

Reprenez le contrôle à l'aide de Linux !

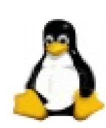
Par Mathieu Nebra (M@teo21)

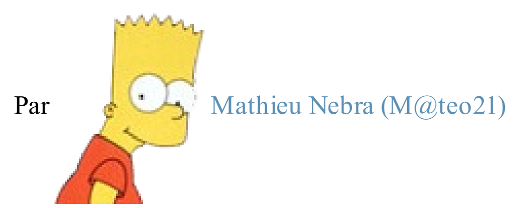


www.siteduzero.com

Dernière mise à jour le 14/10/2012

Sommaire

 Reprenez le contrôle à l'aide de Linux !

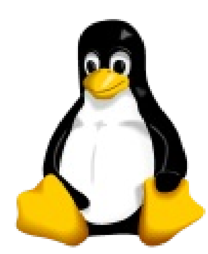
Par  Mathieu Nebra (M@teo21)

Mise à jour : 14/10/2012
Difficulté : Facile Durée d'étude : 1 mois, 15 jours

"*Linux c'est trop compliqué, c'est pour les pros*"
(Dire qu'il y a des gens qui croient ça !)

... Comment ça... c'est ce que vous croyez vous aussi ?
Halte-là, malheureux ! Ne faites pas un pas de plus, vous faites fausse route !

Linux n'est pas compliqué, et je vais vous le prouver.
Vous ne savez pas ce qu'est Linux ? Ce n'est pas grave, c'est un cours pour débutants : les explications commencent dès le premier chapitre !



A qui s'adresse ce cours ?

- Aux utilisateurs de Windows qui veulent découvrir Linux
- Aux linuxiens débutants qui cherchent à mieux maîtriser leur OS
- Aux webmasters qui doivent administrer un serveur dédié sous Linux
- Aux curieux comme vous qui se demandent juste comment Linux fonctionne 

Grâce à Linux, vous avez la possibilité aujourd'hui de *reprendre le contrôle* de votre ordinateur et de découvrir tout un nouveau monde passionnant, le tout sans dépenser un sou !



Ce cours vous plaît ?

Si vous avez aimé ce cours, vous pouvez retrouver le livre "*Reprenez le contrôle à l'aide de Linux*" du même auteur, en vente sur le Site du Zéro, en librairie et dans les boutiques en ligne. Vous y trouverez ce cours adapté au format papier avec une série de chapitres inédits.

Plus d'informations

Partie 1 : Découvrir et installer Linux

 Mais c'est quoi, Linux ?

Linux ? Difficile de ne pas en entendre parler aujourd'hui. À moins d'être restés enfermés dans un bunker antinucléaire coupé de tout lien avec le reste du monde, vous avez forcément rencontré ce mot quelque part sur Internet ou au cours d'une conversation entre amis.

Ce n'est pas étonnant : Linux est partout. Sans Linux, beaucoup de sites web et de programmes n'existeraient pas aujourd'hui.

Mais bon sang, c'est QUOI Linux ?

C'est la question à laquelle ce premier chapitre va répondre, pas de panique.
Nous n'allons rien installer pour le moment, juste découvrir ce que c'est et comment nous en sommes arrivés là… parce que c'est vraiment important ! Alors installez-vous confortablement et commençons par le commencement. ;-)

Un système d'exploitation

Est-ce que vous avez déjà entendu parler de **Windows** ? Non, je ne vous prends pas pour des idiots, je commence juste à partir de zéro ! ;)

Aujourd'hui, la quasi-totalité des PC (si ce n'est plus !) est « livrée avec Windows ». Mais savez-vous ce que cela signifie ?



Oui, moi je sais ! Ça signifie qu'on voit marqué « Windows » au démarrage de l'ordinateur !
Non… ce n'est pas ça ?

Oui allez, disons que c'est un bon début.

En effet, l'une des premières choses que vous voyez lorsque vous allumez votre ordinateur est un écran comme celui de la figure suivante.



Cet écran peut changer en fonction des versions de Windows, mais l'idée est là et vous venez de mentionner le mot clé : Windows se lance **au démarrage** de l'ordinateur.

Le *boot* : démarrage de l'ordinateur

En fait, Windows se lance presque en premier. Si vous regardez bien, vous pouvez constater que c'est quelque chose d'autre qui s'affiche à l'écran au cours des toutes premières secondes. Cette « autre chose » est ce qu'on appelle **l'écran de boot**. Je ne vais pas vous faire de capture d'écran comme pour Windows car cet écran de boot varie beaucoup selon les ordinateurs.

Pourquoi ? Parce qu'il dépend du matériel dont est constitué votre ordinateur. C'est en effet la carte mère qui affiche l'écran de boot. La carte mère est le composant fondamental de tout ordinateur, c'est elle qui fait travailler le processeur, les disques durs, le lecteur de CD-ROM, etc.

On a donc dans l'ordre :

1. écran de boot ;
2. démarrage de Windows.

C'est seulement une fois que Windows est chargé que vous pouvez enfin utiliser vos programmes : jeux, Internet, logiciels de dessin, de mail, de musique…



Mais pourquoi faut-il que Windows se charge d'abord ? Pourquoi ne pourrait-on pas lancer des jeux dès le démarrage de l'ordinateur ?

Parce que… votre ordinateur a besoin d'une sorte de « superlogiciel » qui soit le chef d'orchestre. C'est lui qui doit gérer la mémoire de votre ordinateur, la répartir entre tous les programmes. Il fait le lien entre votre matériel (carte graphique, mémoire, imprimante) et vos logiciels. Et c'est un sacré boulot, croyez-moi ! ;-)

Ce « superlogiciel » s'appelle le **système d'exploitation**. Windows est donc un système d'exploitation.

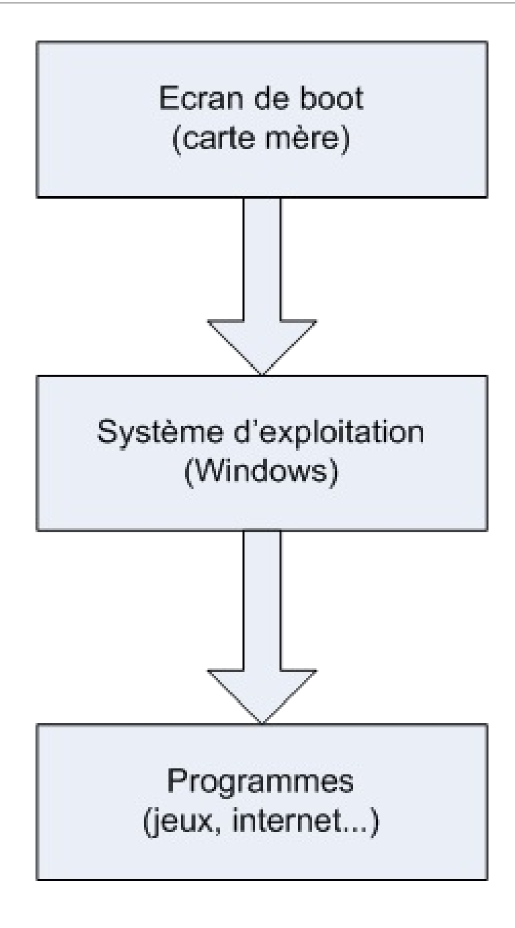


Un système d'exploitation se dit *Operating System* en anglais, que l'on abrège en « OS ». J'utiliserai souvent cette abréviation par la suite, ne soyez donc pas surpris !

Si on résume l'ordre des choses, au final nous avons donc cela :

1. écran de boot ;
2. démarrage du système d'exploitation (Windows) ;
3. lancement des programmes (jeux, Internet, mail…).

Si vous préférez les schémas (je vous conseille de vous y habituer car j'y aurai souvent recours par la suite :)), reportez-vous à la figure suivante.



Linux est un système d'exploitation

Et Linux dans tout ça ?
Rassurez-vous, je ne l'ai pas oublié ! Maintenant que vous savez un peu mieux ce qu'est un système d'exploitation (un OS), je peux vous dévoiler la vérité : **Linux est un système d'exploitation**, au même titre que Windows ou encore Mac OS (pour ceux qui ont un Mac).

Il est réputé entre autres pour sa sécurité et pour ses mises à jour plus fréquentes que Windows ; mais tout cela, vous allez le découvrir petit à petit.
Ce qu'il faut retenir pour le moment est le principe de base de Linux : c'est vous qui contrôlez votre ordinateur. Ce n'est donc pas par hasard si ce cours s'appelle « Reprenez le contrôle à l'aide de Linux ! ». Vous allez enfin comprendre ce que vous faites et donc mieux appréhender le fonctionnement de l'informatique !



Peut-on faire cohabiter deux OS sur son ordinateur ? Je n'ai pas envie de supprimer Windows pour le remplacer par Linux !

Beaucoup de gens croient qu'il faut faire un choix : Linux *ou* Windows. Rien n'est plus faux : vous pouvez très bien avoir deux OS (ou plus !) installés sur votre ordinateur.

Dans ce cas, juste après l'écran de boot, vous aurez un programme appelé **Boot Loader** qui s'affichera pour que vous puissiez faire votre choix. Le *boot loader* dont nous parlerons ici porte le doux nom de **GRUB**.
Celui-ci vous proposera de choisir entre Windows et Linux à chaque démarrage de l'ordinateur. GRUB lancera l'OS par défaut si vous ne faites pas de choix avant quelques secondes. C'est vous qui choisissez l'OS par défaut, bien entendu ; on ne vous impose rien.

Le schéma du démarrage de l'ordinateur change donc un peu, comme le montre la figure suivante.

Ecran de boot
(carte mère)

Boot Loader
(Grub)

Windows

Linux

Programmes Windows
(jeux, internet...)

Programmes Linux
(jeux, internet...)

Comme vous pouvez déjà le voir sur ce schéma, lorsque vous êtes sous Linux, vous utilisez des programmes faits pour Linux et non pas les programmes de Windows. En effet, les programmes Windows ne fonctionnent pas sous Linux et inversement.



Sachez quand même qu'il est possible de faire tourner des programmes Windows depuis Linux à l'aide d'un programme appelé **wine**. Toutefois, même si cela fonctionne bien la plupart du temps, il est davantage préférable d'utiliser des programmes faits pour Linux une fois que vous utilisez cet OS car ils tournent plus rapidement.

Vous devez déjà vous dire : « Aïe, si je ne peux plus utiliser les programmes de Windows auxquels je suis habitué, je ne vais pas m'en sortir ». Pourtant, il ne faut pas longtemps pour s'adapter (deux ou trois jours suffisent, voire même une soirée si vous êtes curieux !) et les programmes sous Linux ont d'énormes avantages :

- ils sont **gratuits** : vous verrez que sous Linux, presque tous les programmes sont gratuits ;
- les logiciels sont **souvent mis à jour** et ce toujours gratuitement ! Vous verrez donc fréquemment vos logiciels préférés évoluer et n'aurez pas à payer trois cents euros pour vous mettre à jour !
- certains de ces logiciels sont **meilleurs** que ceux que l'on trouve sous Windows. D'ailleurs, certains n'existent même pas sous Windows ! Vous découvrirez de nouvelles fonctionnalités et finirez par gagner du temps tout en utilisant plus efficacement votre ordinateur.



Mais pourquoi les programmes sont-ils gratuits ? Ce n'est pas complètement suicidaire financièrement ?

C'est justement ce que je vais vous expliquer maintenant.

La naissance de Linux

Voyons les choses en face.

- **Windows** coûte environ 200 ou 300 euros.
- **Linux** est gratuit, soit 0 euro TTC.

On ne peut pas s'empêcher de se dire : « Mais si c'est gratuit, c'est que cela doit être quelque chose de vite fait et de moindre qualité ! ». Grossière erreur.

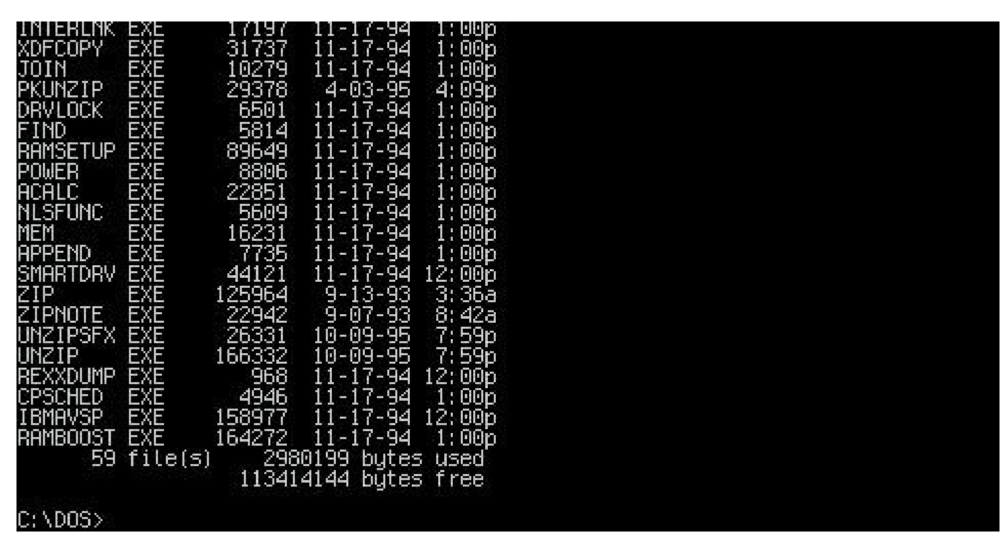
Si Linux est gratuit (comme quasiment tous ses logiciels), il y a des raisons ; pour comprendre, il faut remonter à 1984.

L'informatique en 1984

Nous sommes donc en 1984. À cette époque, l'informatique n'est pas très développée. Microsoft vient de sortir son premier OS : **MS-DOS**. Mais ce dernier est encore loin d'être abouti.



Si vous avez utilisé les premières versions de Windows, vous avez forcément entendu parler de MS-DOS (figure suivante).



MS-DOS, l'ancêtre de Windows



Mais MS-DOS était-il le seul OS existant à l'époque ?

Non ! Il y en avait d'autres mais bien moins connus du grand public.
Celui qui était considéré comme le meilleur s'appelait « **Unix** ». Il était beaucoup plus puissant que MS-DOS mais aussi plus compliqué à utiliser, ce qui explique pourquoi seuls les informaticiens professionnels l'utilisaient.
Il est aussi beaucoup plus ancien : ses origines remontent à 1969 !

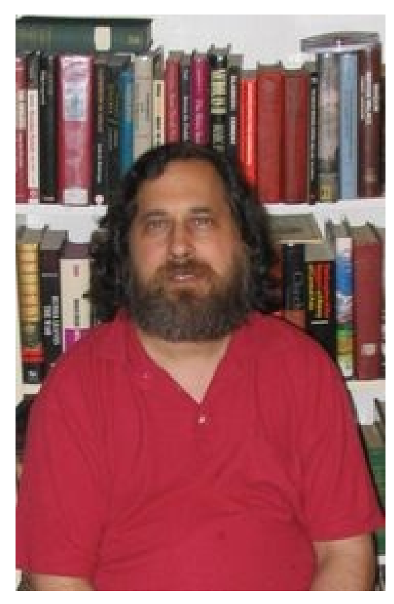
Graphiquement, Unix ressemblait beaucoup à MS-DOS : du texte blanc sur un fond noir. Il faut dire qu'à l'époque les ordinateurs n'étaient pas vraiment capables de faire mieux.

Le projet GNU



Le gnou, emblême de GNU

C'est justement à cette époque, en 1984, que Richard Stallman créa le projet GNU.
Richard Stallman était alors chercheur en intelligence artificielle au MIT. Il voulait créer un nouveau système d'exploitation fonctionnant comme Unix (les commandes restant les mêmes).


Richard Stallman, fondateur du projet GNU

Pourquoi vouloir créer une « copie » d'Unix ?
Parce qu'Unix était payant et devenait de plus en plus cher ! Richard Stallman a voulu réagir en proposant une alternative gratuite : le projet GNU était né.


Bon à savoir : Mac OS X est lui aussi basé sur Unix. En revanche, MS-DOS et Windows sont complètement à part.

GNU est un système d'exploitation libre

GNU ne devait pas seulement être un OS gratuit ; il devait également être « libre ».


Quelle différence ?

Un programme **libre** est un programme dont on peut avoir le code source, c'est-à-dire la « recette de fabrication ».
Au contraire, Windows est un OS **propriétaire** dont le code source est conservé par Microsoft. Imaginez que c'est un peu comme le Coca-Cola : personne ne connaît la recette de fabrication (il y a bien des gens qui essaient de l'imiter, mais bon…). On ne peut donc pas le modifier ou regarder comment il fonctionne à l'intérieur.

Un programme libre est donc la plupart du temps un programme gratuit. Mais c'est aussi un programme qu'on a le droit de copier, modifier, redistribuer.
C'est une véritable idéologie en informatique : des gens pensent qu'il vaut mieux donner le code source des programmes que l'on crée car cela permet le partage des connaissances et aide l'informatique à évoluer plus vite. Le slogan du monde du Libre pourrait être : « L'union fait la force ».


On dit aussi souvent que le programme est « Open Source », car son code source est ouvert ; tout le monde peut le voir.
Il existe quelques légères différences entre un programme « Open Source » et un programme « libre », mais nous n'entrerons pas dans les détails ici.

Pendant ce temps, Linus Torvalds s'amusait

En 1991, **Linus Torvalds**, un étudiant de l'Université de Helsinki (Finlande), entreprend de créer sur son temps libre son propre système d'exploitation.
Ce système a pris le nom de Linux, en référence au nom de son créateur (Linux est la contraction de Linus et Unix).



Linus Torvalds, créateur de Linux

Quel rapport avec GNU ? Eh bien il se trouve que ces deux projets étaient complémentaires : tandis que Richard Stallman créait les programmes de base (programme de copie de fichier, de suppression de fichier, éditeur de texte), Linus s'était lancé dans la création du « cœur » d'un système d'exploitation : le noyau.

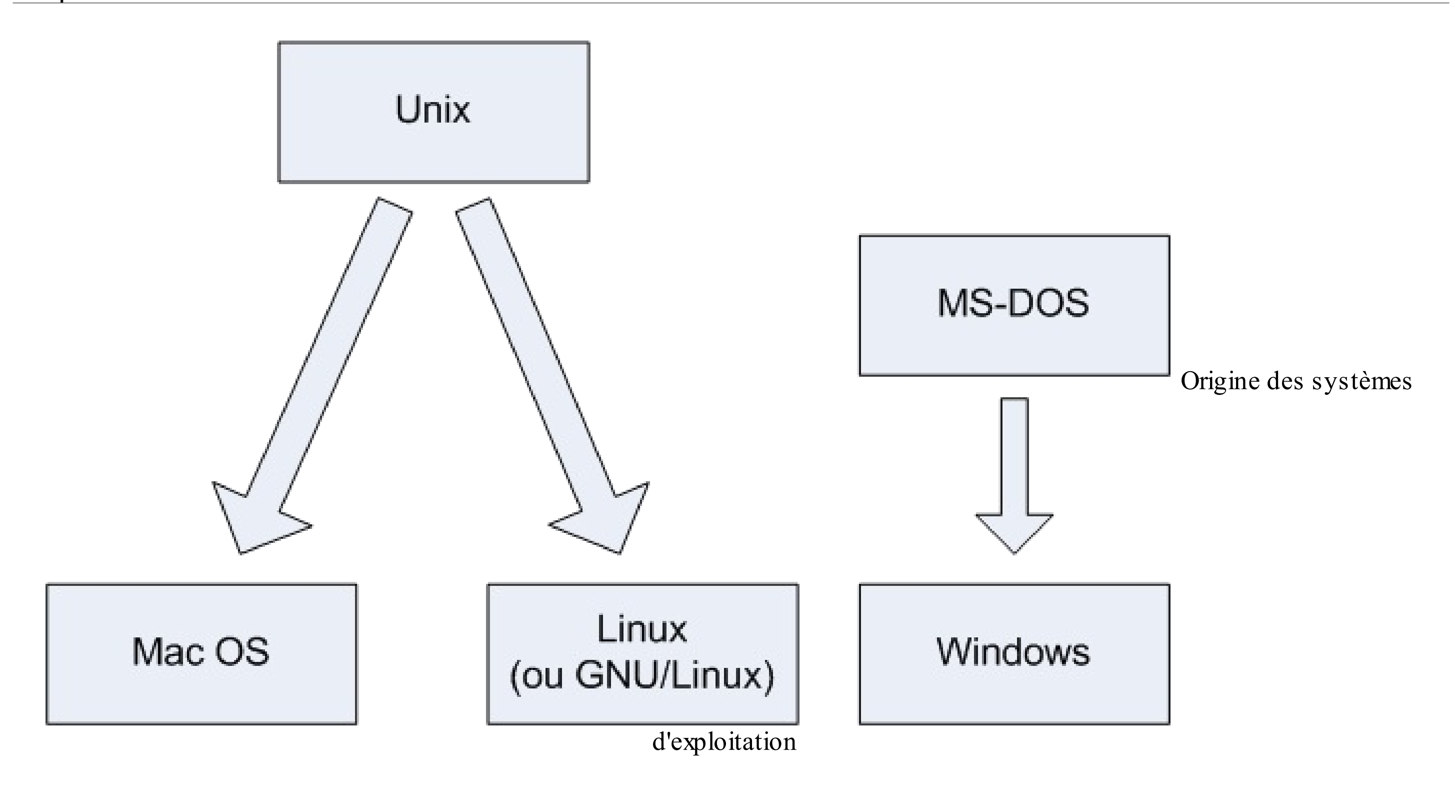
Le projet GNU (programmes libres) et Linux (noyau d'OS) ont fusionné pour créer **GNU/Linux**.



Théoriquement, on doit donc parler de GNU/Linux. C'est toutefois un peu difficile à écrire et prononcer, et par abus de langage, on dit souvent juste « Linux ». C'est donc pour cela que je continuerai à parler de « Linux » dans la suite du livre, même si le nom *politiquement correct* est « GNU/Linux » puisqu'il s'agit de la fusion de deux projets complémentaires.

Résumons avec un schéma !

Ça va, vous n'êtes pas trop embrouillés ?
Je pense qu'une illustration destinée à mettre de l'ordre dans les idées est indispensable ! S'il y a une chose que vous devez retenir, c'est le schéma suivant.



Origine des systèmes d'exploitation

Vous devriez maintenant avoir une meilleure idée de l'origine des trois principaux systèmes d'exploitation qui existent aujourd'hui : Mac OS, Linux et Windows.
Ainsi, Mac OS et Linux sont tous les deux basés sur Unix, l'ancêtre des systèmes d'exploitation, tandis que Windows, issu de MS-DOS, est une branche à part. Globalement, c'est tout ce que vous avez besoin de retenir.

On dit que Mac OS et Linux sont basés sur Unix car ils ont « copié » son mode de fonctionnement. Ce n'est pas péjoratif, bien au contraire : cela fait même honneur à Unix.



Les programmes Linux n'utilisent pas du tout le même code source que ceux d'Unix (celui-ci était d'ailleurs propriétaire, donc privé). Ils ont été complètement réécrits mais fonctionnent de la même manière.

Si je vous ai raconté tout cela c'est parce que j'estime que connaître **l'origine** de Linux est important. Cela vous permettra de comprendre bon nombre de choses par la suite.

Les distributions de Linux

Linux est un système d'exploitation très riche, vous allez pouvoir le constater. On peut y trouver de nombreux logiciels différents et il existe des centaines de façons distinctes de l'installer.

Pour simplifier la vie des utilisateurs et leur permettre de faire un choix, différentes **distributions** de Linux ont été créées. C'est un concept qui n'existe pas vraiment sous Windows. C'est un peu comme la différence entre Windows 7 Familial et Windows 7 Professionnel, mais cela va bien plus loin que ça.

Voici ce qui peut différer d'une distribution à l'autre :

- l'installation : elle peut être très simplifiée comme très compliquée ;
- la gestion de l'installation des programmes. Si elle est bien faite et centralisée, elle peut rendre l'installation de nouveaux logiciels plus simple que sous Windows, comme nous le verrons plus loin !
- les programmes préinstallés sur l'ordinateur (Windows est par exemple livré avec Internet Explorer et Windows Media Player).

En fait, une distribution est en quelque sorte l'emballage de Linux. Le cœur, lui, reste le même sur toutes les distributions.

Quelle que soit la distribution que vous installez, vous obtenez un Linux compatible avec les autres. Certaines distributions sont juste plus ou moins faciles à prendre en main. ;-)

Les différentes distributions existantes

Il existe un grand nombre de distributions Linux différentes. Difficile de choisir, me direz-vous : en effet, la première fois, on ne

sait pas trop pour laquelle opter… surtout que toutes sont gratuites ! Rassurez-vous, je vais vous aider à faire votre choix.

Je ne vais pas dresser la liste de toutes les distributions existantes, mais voici au moins les principales :

- **Slackware** : une des plus anciennes distributions de Linux. Elle existe encore aujourd'hui !
- **Mandriva** : éditée par une entreprise française, elle se veut simple d'utilisation ;
- **Red Hat** : éditée par une entreprise américaine, cette distribution est célèbre et très répandue, notamment sur les serveurs ;
- **SuSE** : éditée par l'entreprise Novell ;
- **Debian** : la seule distribution qui soit gérée par des développeurs indépendants plutôt que par une entreprise. C'est une des distributions les plus populaires.

Comme je vous l'ai dit, quelle que soit la distrib' (abréviation de distribution) que vous choisirez, vous aurez un Linux. Grosso modo, vous aurez « juste » un fond d'écran au premier démarrage et différents logiciels préinstallés (je simplifie ~~un peu~~ beaucoup, mais l'idée est là).

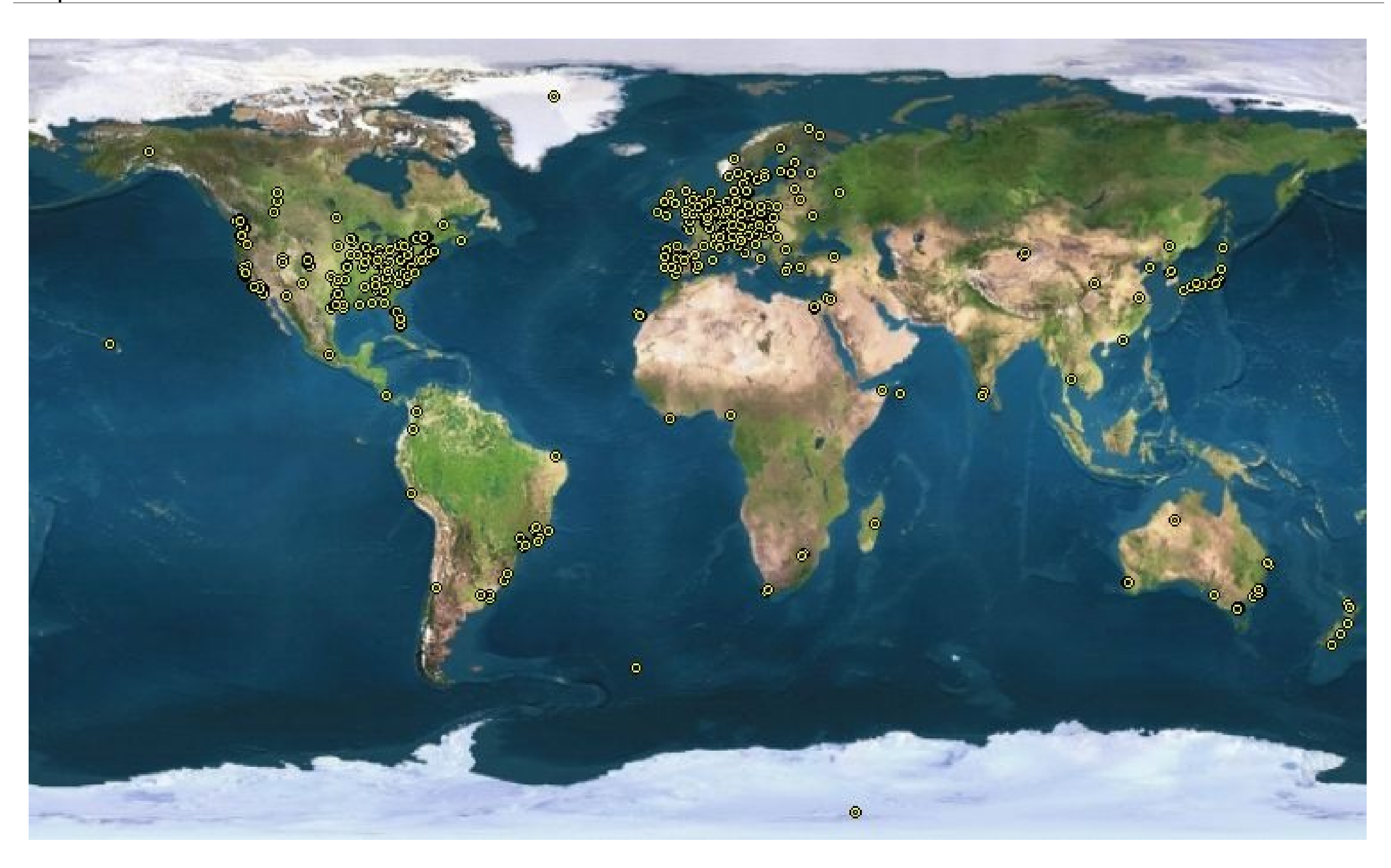
La distribution Debian

Nous, nous allons nous concentrer sur la distribution **Debian** (figure suivante).



Pourquoi Debian ? Tout d'abord parce qu'il nous faut bien faire un choix.
Ensuite parce que c'est la seule distribution qui soit gérée par des gens comme vous et moi (enfin, assez doués en programmation tout de même). Les autres distributions sont gérées par des entreprises, ce qui ne les empêche pas d'être « Open Source » et gratuites, même si nous pouvons également les acheter pour avoir droit à une assistance (hotline…).

Debian est donc la seule distribution éditée par des particuliers bénévoles à travers le monde. Jetez un œil à la carte (figure suivante) pour vous faire une idée.



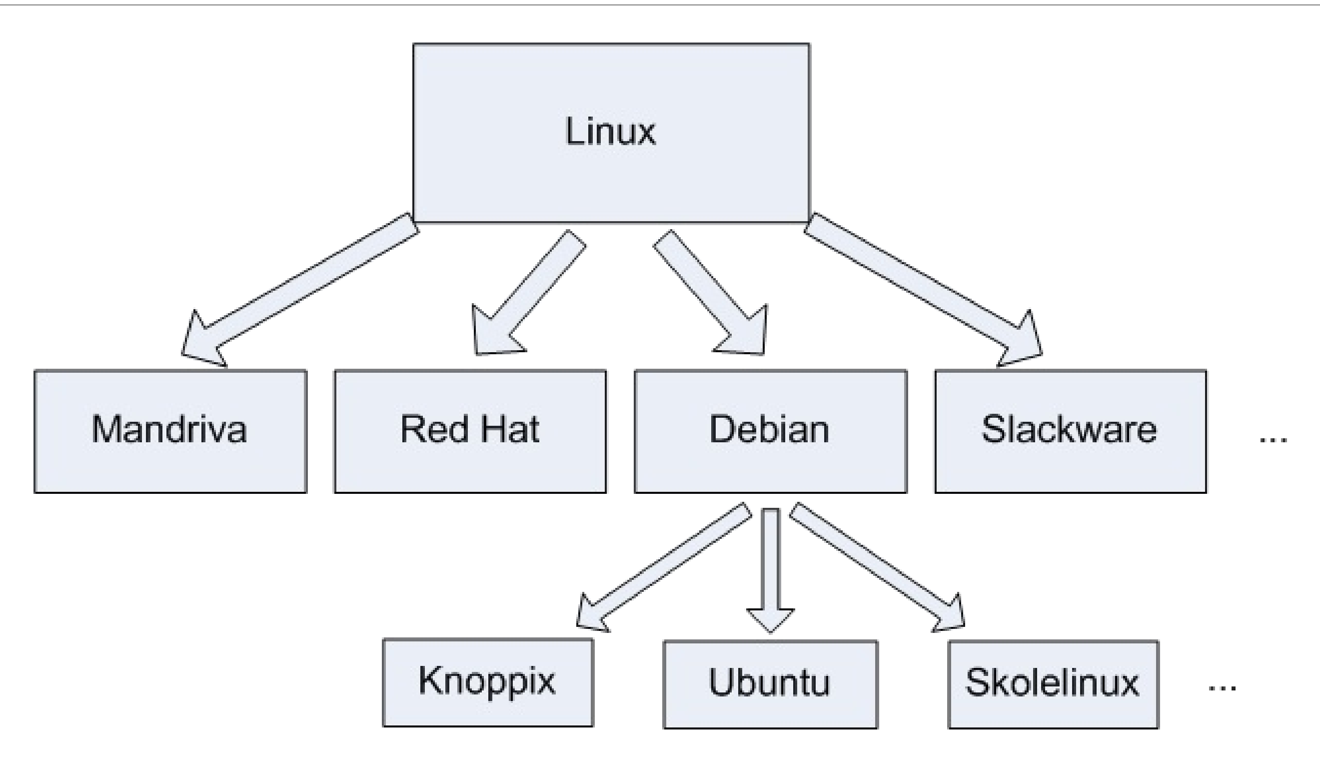
Un autre gros avantage de Debian est le gestionnaire de paquets apt-get. C'est un programme qui gère tous les logiciels installés et qui vous permet de les désinstaller en un rien de temps. D'autre part, tous les logiciels sont centralisés en un même endroit, ce qui fait que vous n'avez pas à parcourir tout le Web pour retrouver un programme.
En fait, vous avez juste à indiquer le nom du logiciel que vous désirez : Debian ira le télécharger et l'installer pour vous. C'est extrêmement simple d'utilisation, je vous montrerai cela !

Debian a tellement de succès que de nombreuses distributions sont basées sur Debian :

- Knoppix ;
- Skolelinux ;
- Ubuntu ;
- …

Ce sont donc des… distributions de distributions. :-)

O.K. : vous trouvez que ce n'est pas clair. Très bien, très bien, je ne vais pas le prendre mal, je vais vous proposer… un nouveau schéma (figure suivante) !



Certaines distributions sont spécialisées. Par exemple, Skolelinux est faite pour être utilisée dans des écoles et est livrée avec de nombreux logiciels éducatifs (gratuits, bien sûr !).

Quant à **Ubuntu** (figure suivante), c'est la distribution qui a créé la surprise. Elle est devenue très populaire en peu de temps. Pourquoi ? Il y a plusieurs raisons à cela.



- Elle est prévue pour le **grand public**, c'est-à-dire des gens comme vous et moi qui n'ont pas envie de se prendre la tête pour utiliser leur ordinateur. Le slogan est *« Linux for human beings »*, ce qui signifie « Linux pour des êtres humains ». Cela veut tout dire.
- **Les mises à jour sont fréquentes** : les développeurs travaillent d'arrache-pied et une nouvelle version de la distribution sort tous les six mois environ, ce qui vous permet de disposer des dernières nouveautés.
- Il y a beaucoup d'utilisateurs, donc **beaucoup de gens pour vous aider** si vous avez des questions (un point à ne pas négliger !).

C'est entre autres pour toutes ces raisons que nous allons utiliser Ubuntu dans cet ouvrage. Nous verrons dans le prochain chapitre comment l'essayer sans l'installer puis comment l'installer tout court si vous êtes conquis et décidés.

En résumé

- Le **système d'exploitation** est l'outil qui fait le lien entre votre machine et vos programmes.
- Windows, Mac OS et Linux sont les systèmes d'exploitation les plus connus.
- Linux a la particularité d'être **libre**, c'est-à-dire que son code source (sa recette de fabrication) est ouvert : tout le monde peut le consulter. Par opposition, le code source qui a permis de concevoir Windows et Mac OS est fermé, on dit que ce sont des systèmes d'exploitation propriétaires.
- Il existe de nombreuses variantes de Linux, que l'on appelle **distributions**.
- **Ubuntu** est une des distributions les plus populaires à l'heure actuelle. C'est celle que nous utiliserons tout au long de cet ouvrage. Il s'agit d'un dérivé de la distribution Debian.

 Téléchargez Linux, c'est gratuit !

Le premier chapitre vous aura permis, je l'espère, de vous mettre un peu dans le bain du monde de Linux.
Nous avons vu ce qu'est Linux, comment est né ce dernier et ce que sont les distributions.

Je vous ai dit en particulier que, sous Linux, nous avons énormément de choix. Il existe en effet de très nombreuses distributions qui proposent des versions différentes de Linux, fort heureusement toutes compatibles entre elles. Ici, j'ai choisi de vous présenter Ubuntu car c'est une distribution très populaire et facile à utiliser.

Comme promis, c'est dans ce chapitre que nous passons à la pratique. Nous allons dans un premier temps découvrir ce que sont les gestionnaires de bureau et choisir en conséquence la version d'Ubuntu qui nous convient le mieux (eh oui, on va encore devoir faire un choix !).

Les deux visages de Linux

À quoi ressemble Linux ?
Si vous vous êtes déjà posé cette question, vous avez peut-être pu observer de nombreuses captures d'écran, toutes très différentes les unes des autres. Il faut dire que Linux est très personnalisable, mais ça je crois que vous commencez à le comprendre à force de le lire.

Si vous n'avez jamais vu de capture d'écran, ou si vous n'avez jamais vraiment fait attention, vous vous posez sûrement cette question…



Linux, c'est plus joli ou moins joli que Windows ?

Il n'y a pas de bonne réponse : cela peut être très beau comme très moche. Comme on peut très facilement changer l'apparence de son Linux, vous arriverez sans problème à trouver une apparence qui vous convient. C'est un peu comme sous Windows, où, vous le savez peut-être, on peut changer l'apparence du système. La différence, c'est que sous Linux la personnalisation va plus loin qu'un simple changement de couleurs. Nous allons voir cela un peu plus en détail.

Ce que je veux que vous sachiez ici c'est que quelle que soit la distribution il existe deux façons d'utiliser Linux :

- en mode console (équivalent à DOS) ;
- en mode graphique (équivalent à Windows).

En mode console

Le mode « console » est un mode qui a tendance à faire peur aux petits nouveaux ; et pour cause, il n'a pas une tête très accueillante : jugez plutôt (figure suivante).

```
ubuntu@ubuntu:~/Desktop$ ls
Examples  ubiquity-kdeui.desktop
ubuntu@ubuntu:~/Desktop$ cd Examples
ubuntu@ubuntu:~/Desktop/Examples$ ls
book                      logo-Kubuntu.png          oo-maxwell.odt
book-toc.html             logo-Ubuntu.png           oo-payment-schedule.ods
Experience ubuntu.ogg     oo-about-these-files.odt  oo-presenting-kubuntu.odp
fables_01_01_aesop.spx    oo-about-ubuntu-ru.rtf    oo-presenting-ubuntu.odp
gimp-ubuntu-splash.xcf    oo-access.odt             oo-trig.xls
kubuntu-leaflet.png       oo-cd-cover.odg           oo-welcome.odt
logo-Edubuntu.png         oo-derivatives.doc        ubuntu Sax.ogg
ubuntu@ubuntu:~/Desktop/Examples$ pwd
/home/ubuntu/Desktop/Examples
ubuntu@ubuntu:~/Desktop/Examples$ w
 22:44:02 up 15 min,  7 users,  load average: 0,07, 0,29, 0,26
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
ubuntu   tty1     -                22:30    0.00s  2.93s  0.02s w
ubuntu   tty2     -                22:30   15:25m  0.17s  0.14s -bash
ubuntu   tty3     -                22:30   15:25m  0.15s  0.12s -bash
ubuntu   tty4     -                22:30   15:25m  0.17s  0.14s -bash
ubuntu   tty5     -                22:30   15:25m  0.15s  0.13s -bash
ubuntu   tty6     -                22:30   15:25m  0.17s  0.15s -bash
ubuntu   :0       -                22:30   ?xdm?  50.06s  0.15s /bin/sh /usr/bi
ubuntu@ubuntu:~/Desktop/Examples$ _
```

Le mode console. Oui oui, il s'agit bien d'un Linux du XXIe siècle !

En console, le fond est généralement noir (mais ce n'est pas une règle). Il est cependant possible d'utiliser de la couleur.
Autre point important : en console, pas de souris. Tout se fait au clavier.



Quelle horreur ! Je vais être obligé d'utiliser ça ?

Pas du tout ! Comme je vous l'ai dit plus haut, deux modes sont disponibles. La console n'est qu'un des deux « visages » de Linux ; il existe aussi un mode graphique (encore heureux).

Vous vous demandez à coup sûr ce que fait encore le mode console dans Linux. C'est vrai quoi, si nous avons inventé des écrans plats gigantesques pouvant afficher des milliards de couleurs avec un contraste de 10000:1, ce n'est pas pour retomber à l'âge de pierre !
Et pourtant… la console est un outil très puissant, pratiquement incontournable. Elle est toujours utilisée aujourd'hui par les linuxiens dont vous ferez bientôt partie.

Comment puis-je être aussi sûr de moi ?… Parce que je compte bien vous en expliquer le fonctionnement ! En partant de zéro bien sûr.

En mode graphique

Le mode graphique semble beaucoup plus accueillant pour quelqu'un venant de Windows. En fait, cela ressemble un peu à ce dernier : il y a des fenêtres et on clique sur des croix pour les fermer. Standard, quoi. ;-)

Le truc… c'est qu'il y a plusieurs modes graphiques. Tous les modes graphiques sont basés sur un programme appelé **X** (voilà un nom court et facile à retenir). X est en fait la brique de base du mode graphique sous Linux.

Par-dessus X vient se greffer un programme appelé le **gestionnaire de bureau**. Le rôle du gestionnaire de bureau est de gérer les fenêtres, leur apparence, leurs options, etc.



Le concept de gestionnaire de bureau n'existant pas sous Windows, il s'agit donc de quelque chose nouveau pour vous.
Certes, sous Windows on peut changer l'apparence (le « skin »), mais cela s'arrête là. Le bureau reste le même, il y a


toujours une barre des tâches avec le menu *Démarrer*, toujours un *Poste de Travail*, etc.
Sous Linux en revanche le fonctionnement peut radicalement changer d'un gestionnaire de bureau à l'autre !

Présentation des principaux gestionnaires de bureau

Je ne vais pas vous faire la liste de tous les gestionnaires de bureau qui existent (ils sont vraiment nombreux !), mais commencez déjà par retenir ces quatre-là, qui sont probablement les plus célèbres :

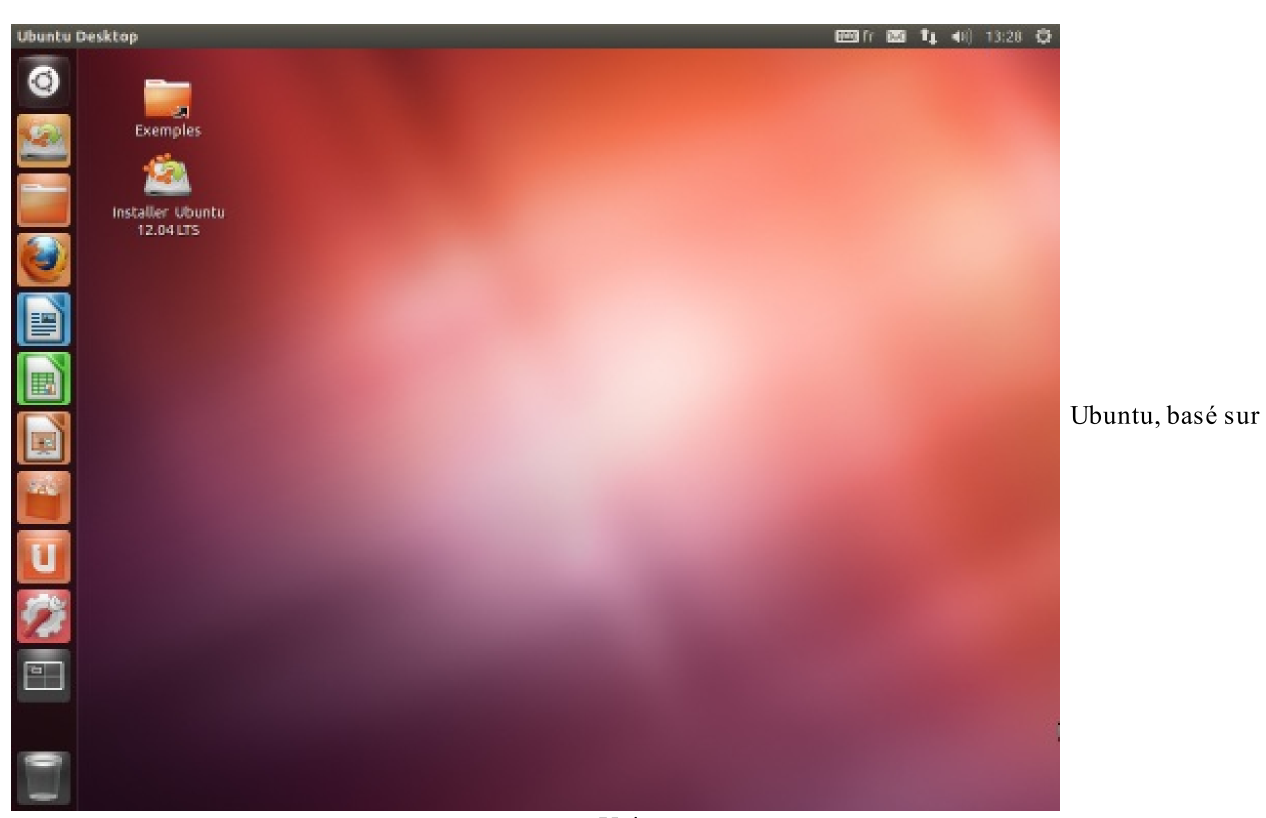
- Unity ;
- Gnome ;
- KDE ;
- XFCE.

Unity, Gnome et KDE sont de « gros » gestionnaires de bureau, très riches en fonctionnalités. XFCE est une alternative plus légère qui peut fonctionner sur des PC plus anciens.
Si votre PC peut faire tourner Windows, il ne rencontrera aucun problème avec Unity ou KDE. En revanche, s'il est un peu plus ancien, vous devrez peut-être vous pencher sur XFCE, moins gourmand en ressources.

Pourquoi est-ce que je vous parle des gestionnaires de bureau ? Parce qu'il va falloir en choisir un ! Ubuntu vous permet en effet de sélectionner un gestionnaire de bureau. Par défaut, Ubuntu est livré avec Unity, mais vous pouvez en changer à tout moment. Il existe d'ailleurs plusieurs versions préconfigurées d'Ubuntu (même si elles ne sont pas officiellement supportées) comme Kubuntu, Xubuntu...

Voici un descriptif illustré de chacun d'entre eux.

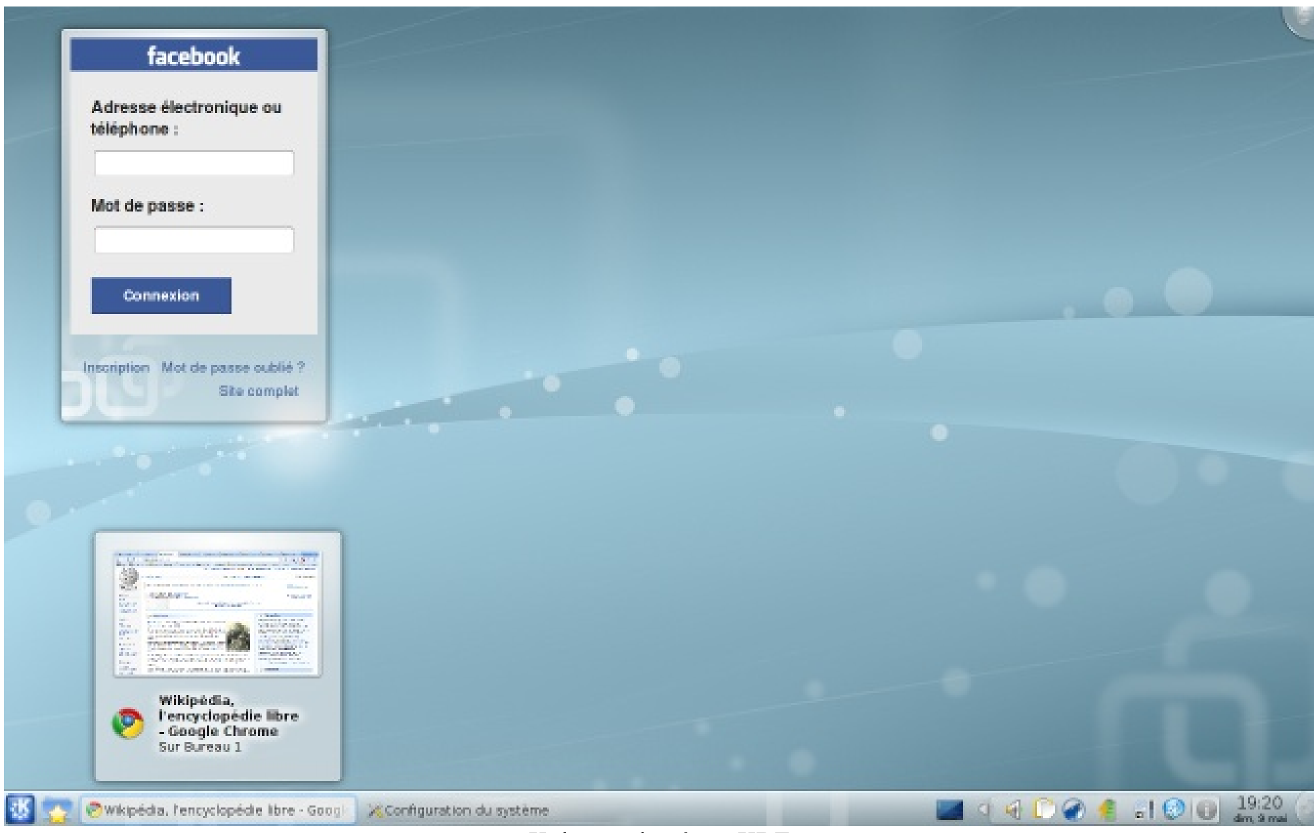
- **Unity** — Nom de la distribution Ubuntu : **Ubuntu** (figure suivante). Unity est un des gestionnaires de bureau les plus utilisés, de part son intégration au sein d'Ubuntu. Le système est contrôlé par un menu présent en haut de l'écran, un peu à la manière de Mac OS (mais la ressemblance s'arrête là).
  À la base, Ubuntu proposait un gestionnaire de bureau appelé Gnome. Au fil du temps, Ubuntu a été décliné en plusieurs versions pour que chacun puisse choisir son gestionnaire de bureau favori. Mais depuis la version 11.04 d'Ubuntu, Gnome a laissé place à Unity, le nouveau gestionnaire de bureau par défaut.



Ubuntu, basé sur Unity

- **KDE** — Nom de la distribution Ubuntu : **Kubuntu** (figure suivante). KDE est un autre poids lourd des gestionnaires de bureau. C'est un des plus anciens, et il a énormément évolué au fil des années. Très complet lui aussi, il possède une sorte de « barre des tâches » en bas de l'écran, semblable à celle de Windows (le menu « K » étant ici l'équivalent du menu « Démarrer »). À cause de cette vague (j'ai bien dit vague !) ressemblance avec Windows, on a tendance a être plus

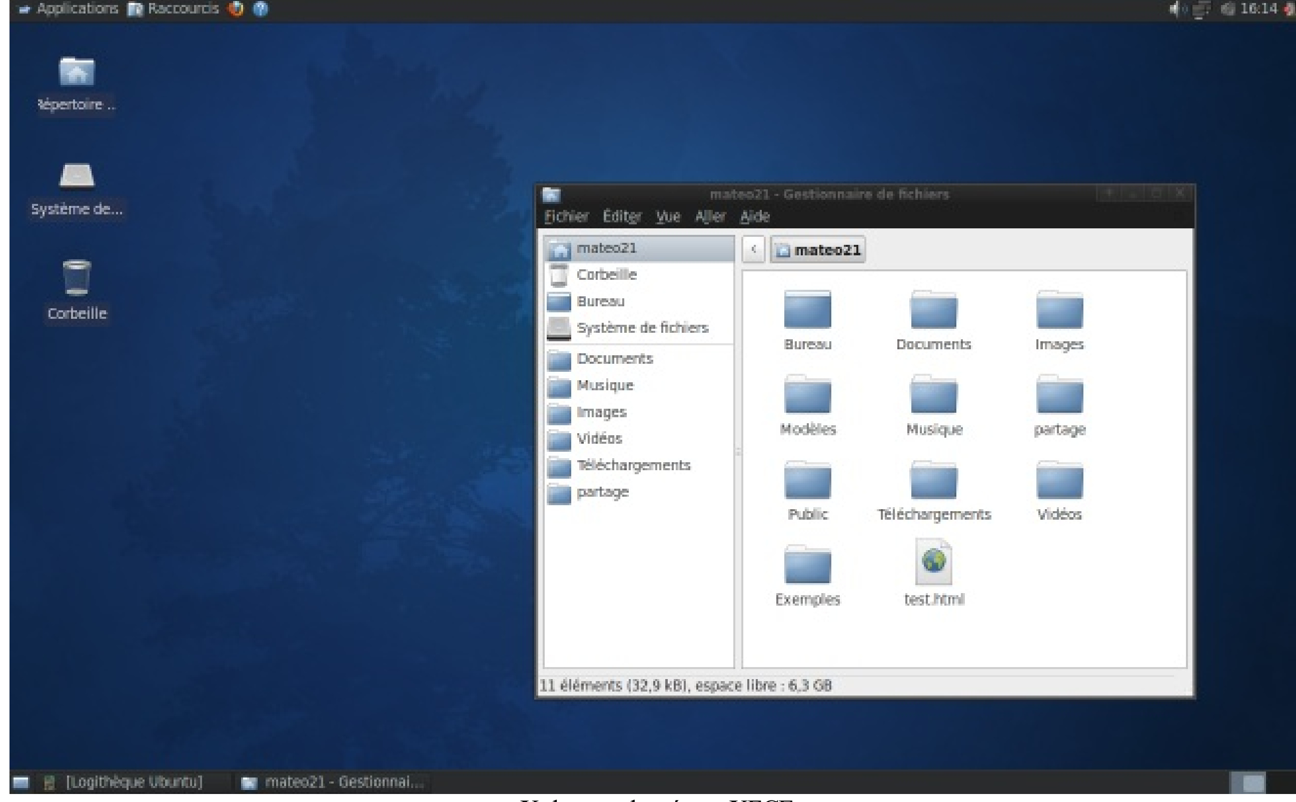
attiré par KDE quand on débute sous Linux.



Kubuntu, basé sur KDE

- **XFCE** — Nom de la distribution Ubuntu : **Xubuntu** (figure suivante). XFCE est une alternative plus légère que Unity et KDE. Il est donc en toute logique un peu moins pourvu en fonctionnalités. Ça ne veut pas dire qu'il est simplet, loin de là ; il se révèle très agréable à utiliser.

  Son apparence est proche de celle de Unity mais peut aussi tout à fait ressembler à KDE.

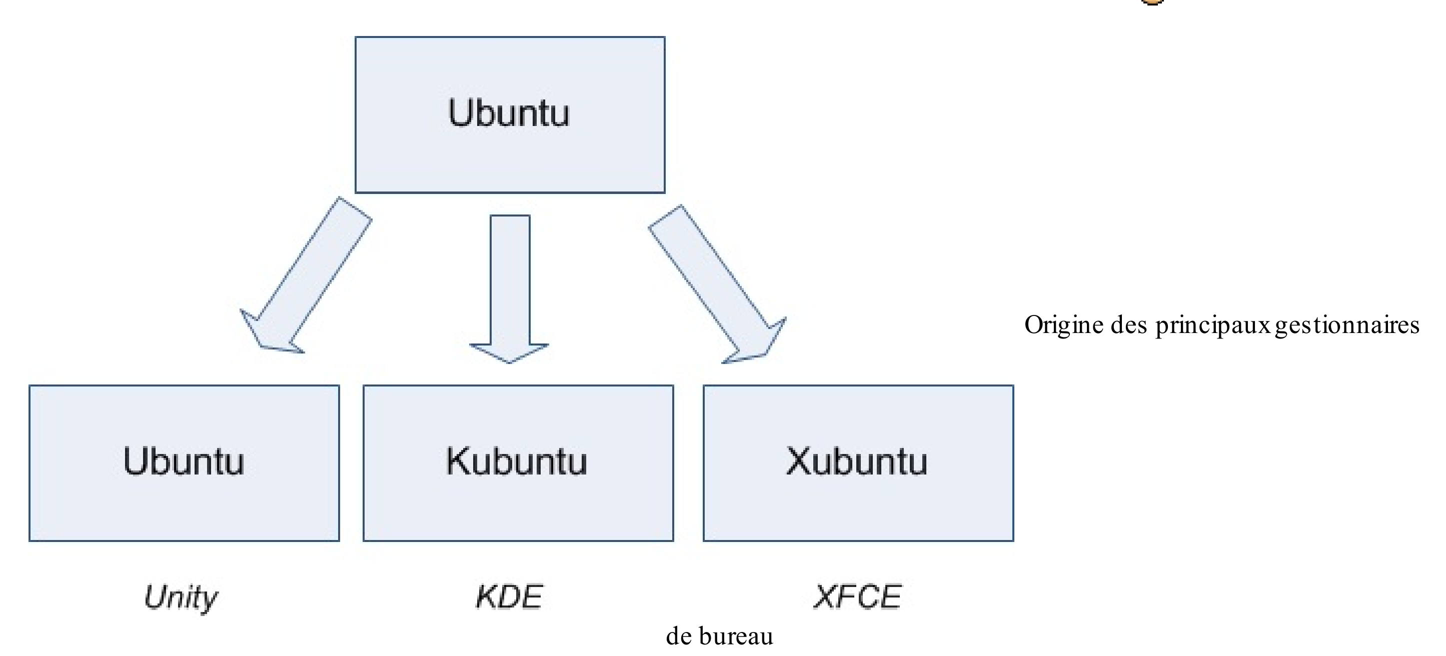


Xubuntu, basé sur XFCE

La première version d'Ubuntu était basée sur le gestionnaire de bureau Gnome. Le succès d'Ubuntu grandissant, les utilisateurs de KDE et de XFCE ont voulu eux aussi voir des versions d'Ubuntu basées sur leur gestionnaire de bureau favori. De là sont

nées Kubuntu (basée sur KDE) et Xubuntu (basée sur XFCE). Désormais, Ubuntu est basé sur Unity par défaut, mais vous pouvez toujours installer Gnome par la suite si vous le désirez.

Tiens, cela fait longtemps que je n'ai pas fait de schéma et cela me manque… voyez la figure suivante. 



Origine des principaux gestionnaires de bureau

C'est tout ce que vous avez besoin de retenir pour le moment.
Ubuntu, Kubuntu et Xubuntu sont strictement identiques. Seul le gestionnaire de bureau installé par défaut change. Quand on parle d'Ubuntu, on fait donc généralement référence à toutes les versions d'Ubuntu à la fois.



Le choix du gestionnaire de bureau n'est pas définitif. On peut sans problème avoir plusieurs gestionnaires de bureau installés à la fois. Il vous faudra alors choisir au démarrage, lorsque l'on vous demande votre identifiant et votre mot de passe, le gestionnaire de bureau que vous souhaitez utiliser. Vous pourrez donc tester et installer d'autres gestionnaires de bureau par la suite.
Sachez d'ailleurs qu'il en existe des moins répandus et qui permettent d'avoir un bureau vraiment très différent de Windows.

Enfin, une information importante à retenir : tous ces gestionnaires de bureau sont compatibles entre eux. Les programmes fonctionnent donc tous quel que soit le gestionnaire de bureau que vous utilisez.

Alors… Unity, KDE ou XFCE ? Ubuntu, Kubuntu ou Xubuntu ?

À vous de choisir. Votre première expérience sous Linux sera différente selon que vous choisissez Ubuntu, Kubuntu ou Xubuntu. Il sera toujours possible par la suite de changer de gestionnaire de bureau comme je vous l'ai dit précédemment. Par conséquent, si vous faites une « erreur », ce n'est pas un drame.

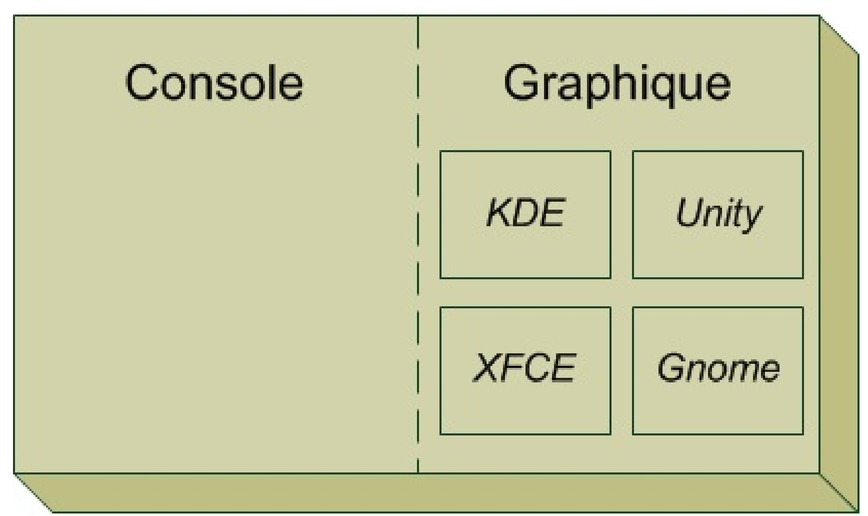
De mon côté aussi, il va bien falloir que je fasse un choix. Et là, c'est délicat. Il y a des utilisateurs de Linux qui ne jurent que par Unity, d'autres par Gnome, d'autres que par KDE… et d'autres que par XFCE. Je ne veux rien vous imposer.

Pour ma part, je vais ici faire le choix de **Unity** (donc d'Ubuntu). Il y a diverses raisons à cela, la première étant que Unity est le gestionnaire de bureau par défaut d'Ubuntu. Les autres distributions (Kubuntu, Xubuntu…) ne sont que des déclinaisons. En pratique, vous aurez aussi plus de chances de tomber sur quelqu'un qui utilise Ubuntu sous Unity le jour où vous aurez besoin d'aide.

Ne vous focalisez pas trop sur le gestionnaire de bureau. Ce qui compte en fait c'est que la console reste strictement identique, et ce que vous utilisiez Unity, KDE ou XFCE. En effet, la console est une « constante » : elle ne change pas d'un Linux à l'autre. Je pourrai donc par la suite vous expliquer le fonctionnement de la console quel que soit le gestionnaire de bureau que vous aurez choisi.

Schéma résumé à retenir

Allez, un petit schéma (figure suivante) pour être sûr que cela rentre et nous pourrons passer à la suite.



La console reste la même quel que soit le gestionnaire de bureau

Ce qu'il faut donc retenir, c'est que Linux peut être utilisé dans deux modes différents : console ou graphique.
Le fonctionnement de la console est le même d'un Linux à un autre ; par contre, l'aspect graphique peut radicalement changer selon le gestionnaire de bureau que l'on choisit : KDE, Unity, XFCE, etc.

Télécharger et graver le CD

Intéressons-nous maintenant au concret : comment obtenir Ubuntu sur CD pour le tester et peut-être l'installer ?

Vous avez deux solutions :

- vous pouvez télécharger Ubuntu vous-mêmes… ;
- … ou bien commander des CD par la poste.

Nous allons commencer par voir comment télécharger Ubuntu.

1/ Récupérer l'ISO

Linux se télécharge sous la forme d'un gros fichier `.iso` d'environ 700 Mo. Ce fichier correspond à l'image d'un CD et vous permet donc de graver un CD complet de Linux.

La première étape consiste à récupérer le fichier ISO. Là, tout dépend si vous avez choisi Ubuntu, Kubuntu ou Xubuntu, car ce n'est pas le même ISO.

Rendez-vous sur une de ces pages en fonction de la version d'Ubuntu que vous désirez. Si vous hésitez, je vous conseille de prendre Ubuntu.

- Télécharger Ubuntu
- Télécharger Kubuntu
- Télécharger Xubuntu

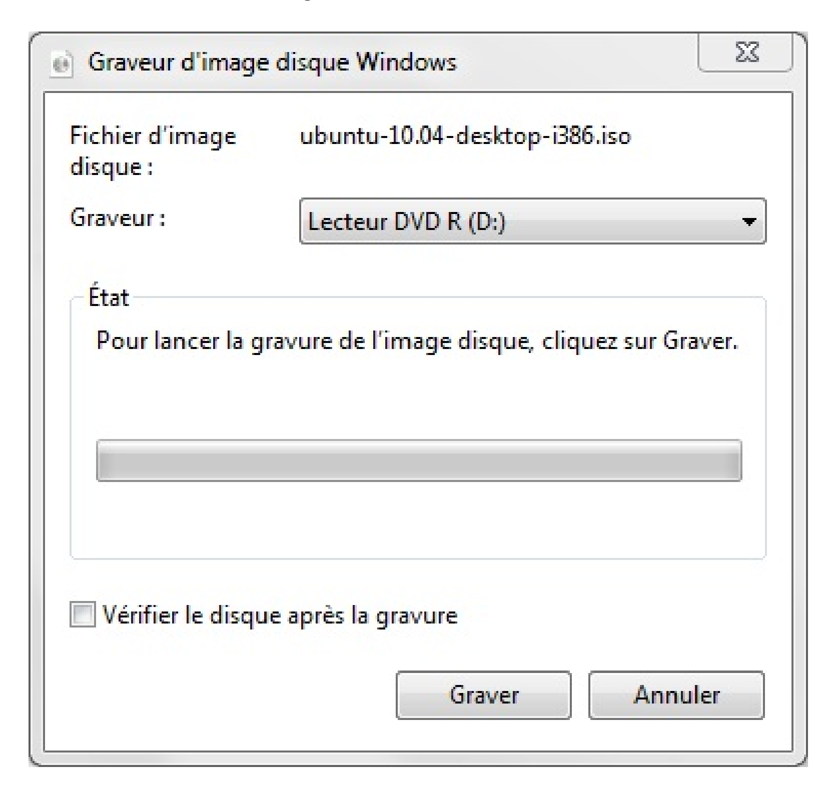
Sur la page, cliquez tout simplement sur le lien de téléchargement. Vous allez récupérer un fichier `.iso`.

2/ Graver le CD

Il vous faut maintenant graver le gros fichier `.iso` que vous venez de télécharger.

Sous Windows 7

Si vous avez Windows 7, un outil de gravure d'images disque `.iso` est déjà inclus. Il vous suffit de double-cliquer sur le fichier `.iso`, ce qui aura pour effet d'ouvrir la fenêtre de la figure suivante.



Insérez un CD vierge dans votre graveur et cliquez tout simplement sur « Graver ».

Sous d'anciennes versions de Windows

Il vous faut un logiciel de gravure pour effectuer l'opération car les versions antérieures à Windows 7 ne savent pas graver les images disque.

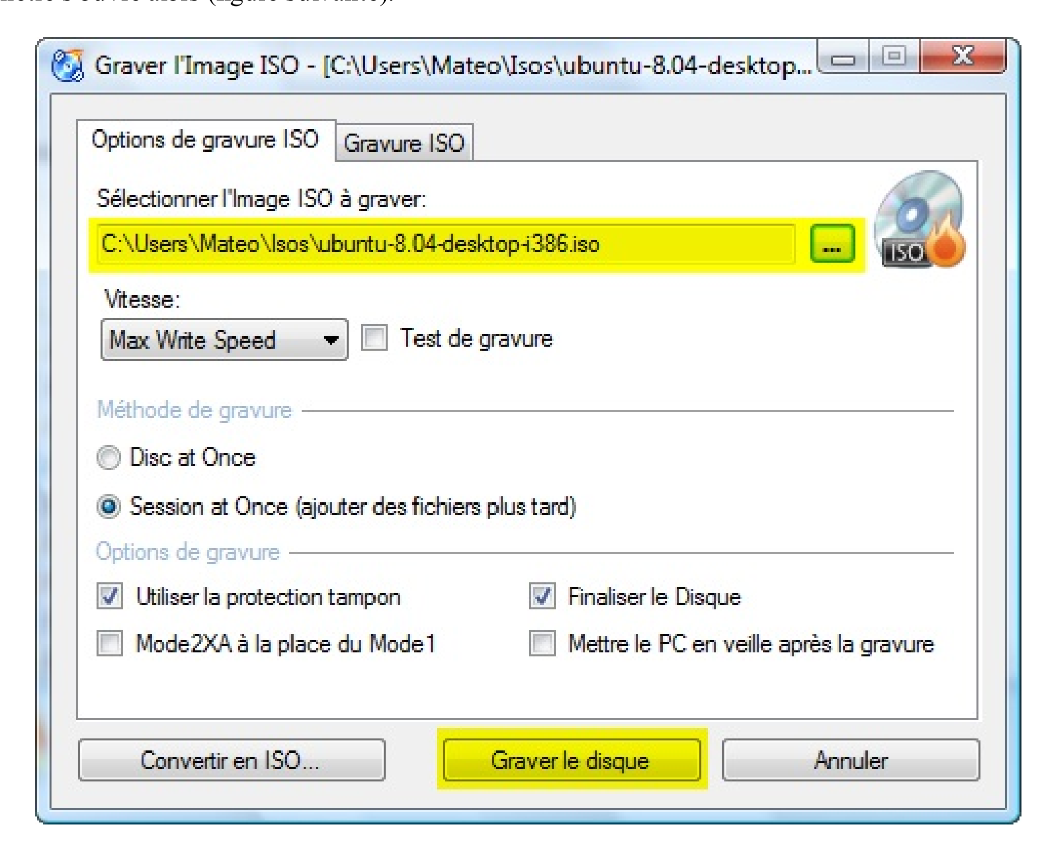
Si vous avez déjà un programme comme Nero ou Easy CD Creator et que vous savez comment graver un ISO, c'est très bien. Sinon, je vais vous montrer comment faire à l'aide du logiciel de gravure gratuit **CDBurnerXP Pro**.

Tout d'abord, commencez par télécharger CDBurnerXP Pro.
Le logiciel est en français.

Insérez un CD-R (CD vierge) dans votre graveur.
Lancez ensuite le logiciel CDBurnerXP Pro et, lors du démarrage, cliquez sur « Créer un CD/DVD de données », comme vous le montre la figure suivante.



La fenêtre principale s'ouvre. Allez dans le menu `Fichier → Graver le disque à partir du fichier ISO`. Une nouvelle fenêtre s'ouvre alors (figure suivante).



Commencez par indiquer en haut où se trouve le fichier ISO que vous venez de télécharger.
Vous pouvez graver à vitesse maximale, mais cela peut parfois provoquer des erreurs, comme une coupure pendant l'installation de Linux. Si vous êtes plutôt prudents, je vous recommande de réduire la vitesse de gravure (vous pouvez mettre 2X ou même 1X).

Cliquez ensuite sur le bouton « Graver le disque », patientez quelques minutes ; c'est prêt ! :-)

Eh bien voilà, ce n'était pas bien compliqué.
Vous avez maintenant un CD d'Ubuntu flambant neuf, gratuit, légal, qui n'attend que d'être essayé.

Nous verrons justement dans le chapitre suivant comment tester Linux.

En résumé

- Linux peut être utilisé de deux façons différentes : en mode console ou en mode graphique.
- Le **mode console** est puissant mais plus délicat à apprivoiser. Les débutants préfèrent l'éviter au début.
- Le **mode graphique** est similaire à ce que vous connaissez sous Windows et Mac OS : on y manipule des fenêtres avec une souris.
- Il existe plusieurs variantes du mode graphique que l'on appelle **gestionnaires de bureau** : Unity, KDE, XFCE...
- Pour obtenir Ubuntu, il suffit de télécharger gratuitement une **image de disque** (format `.iso`) et de la graver sur un CD.

 Tester et installer Ubuntu

Nous y voici enfin. Dans ce chapitre, vous allez peut-être pour la première fois de votre vie voir à quoi ressemble Linux sur votre ordinateur et découvrir qu'en fait… c'est beaucoup plus simple que vous ne le pensiez.

Saviez-vous qu'on peut tester Linux sur son ordinateur sans rien installer sur son disque dur ? On va justement voir maintenant comment faire cela.
Ensuite, je vous montrerai comment installer Linux sur votre disque dur si vous êtes décidés. ;-)

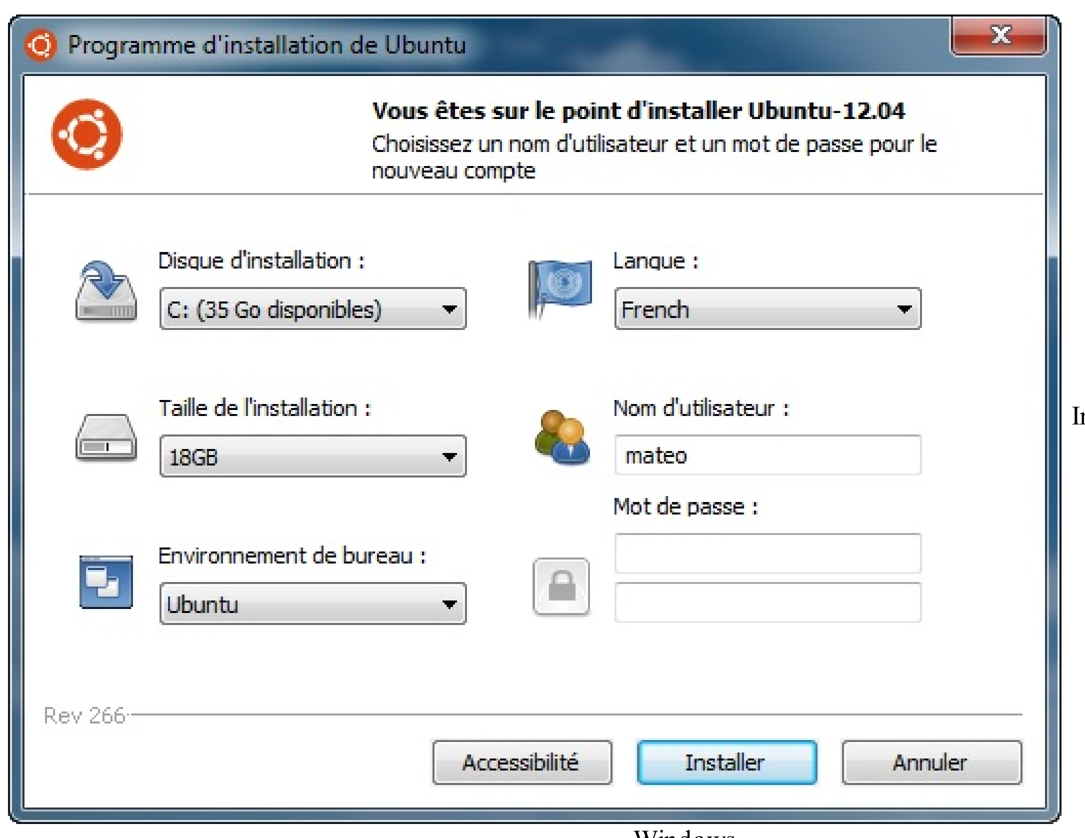
Installation de Linux depuis Windows

Depuis quelque temps, l'installation de Linux a fait des progrès étonnants pour faire en sorte d'être la plus simple possible.

Vous avez aujourd'hui deux possibilités pour installer Linux :

- vous pouvez l'installer **depuis Windows**, dans ce cas l'installation sera un peu particulière mais Linux fonctionnera parfaitement ;
- vous **redémarrez votre ordinateur** avec le CD de Linux dans votre lecteur, ce qui est la méthode la plus classique pour l'installer. On verra comment faire cela dans la suite de ce chapitre.

L'installation de Linux depuis Windows est une petite prouesse technologique qui vous apporte un certain nombre d'avantages. Il vous suffit de télécharger le programme d'installation spécial Windows et de suivre les instructions (figure suivante).



Installation de Linux depuis Windows

Dans cette fenêtre, choisissez la quantité d'espace disque que vous voulez réserver à Ubuntu (en Go). Choisissez aussi un nom d'utilisateur et un mot de passe, puis cliquez sur « Installer ».

L'installation se fait de manière classique depuis Windows. Une fois que cela sera fait, vous pourrez redémarrer votre ordinateur et lancer Ubuntu (il faudra faire un choix au démarrage).
Lors du premier lancement, Ubuntu devra finaliser l'installation, après quoi ce sera bon : vous serez enfin sous Linux.



Un autre gros avantage de cette méthode est que vous pourrez ensuite désinstaller Ubuntu le plus simplement du monde en allant dans… « Ajout / Suppression de programmes » du panneau de configuration de Windows !

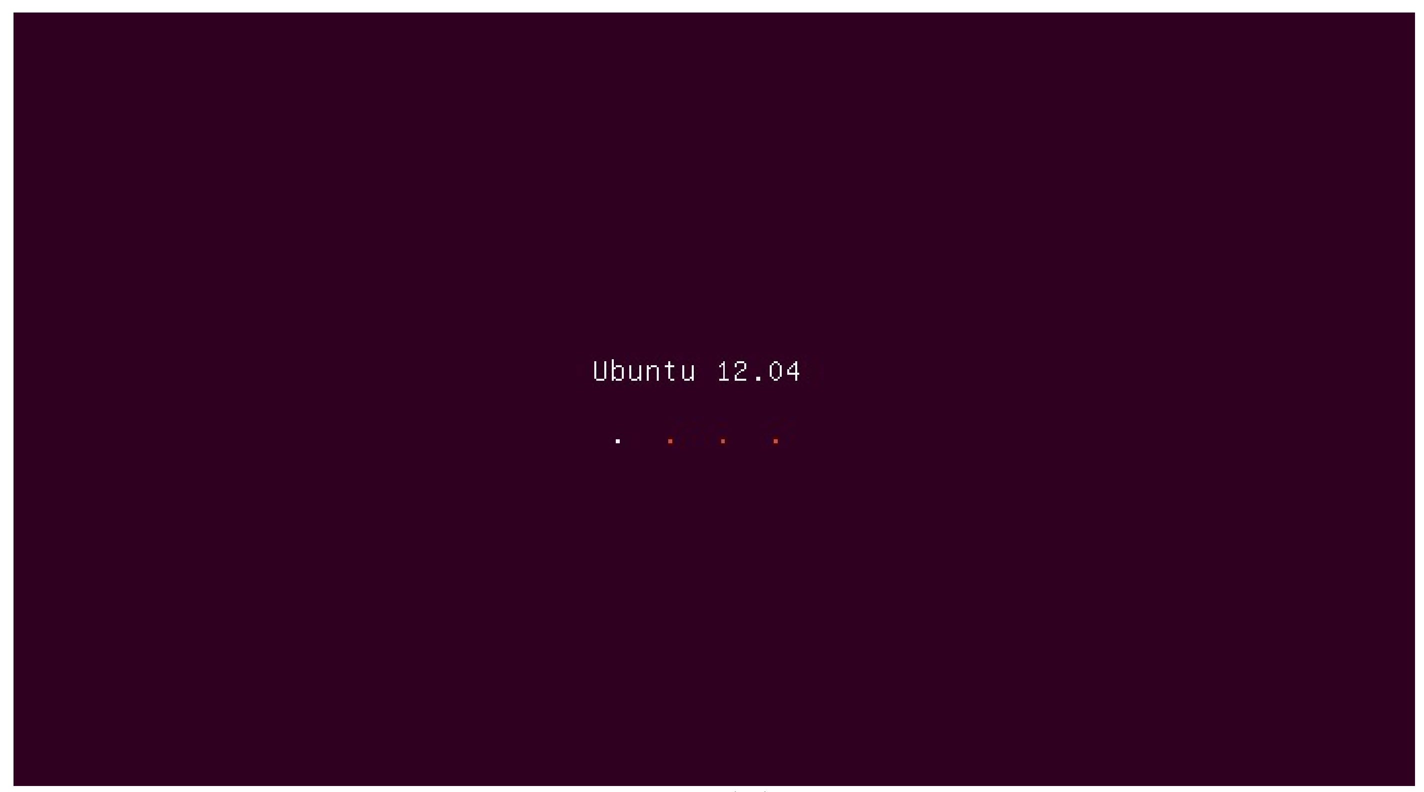
Cette méthode a toutefois des défauts. Ubuntu sera un peu moins performant (car il sera installé **dans** Windows) et nécessitera plus de mémoire vive (512 Mo).

Dans la mesure du possible, je vous conseille d'installer Ubuntu en utilisant la « vraie » méthode classique basée sur un CD d'installation. Nous allons justement voir comment fonctionne l'installation classique ci-dessous.

Premier démarrage d'Ubuntu

Je suppose que pour le moment vous êtes sous votre système d'exploitation habituel, c'est-à-dire Windows (ou Mac OS).
Je vais vous demander de mettre le CD d'Ubuntu dans votre lecteur CD… lààà… voilà, très bien.

Maintenant, redémarrez votre ordinateur.
Vous allez voir Windows s'éteindre, puis l'ordinateur redémarrer. Cette fois, il devrait afficher l'écran de chargement d'Ubuntu, visible sur la figure suivante.



Démarrage d'Ubuntu

Si vous voyez cela, c'est très bien ! Cela signifie que votre ordinateur a démarré sur votre CD qui contient Linux au lieu de démarrer sur le disque dur sur lequel est installé Windows.


Euh… moi j'ai mis le CD dans le lecteur, j'ai redémarré, et pourtant ça a lancé Windows quand même !
Je dois jeter mon PC par la fenêtre ?

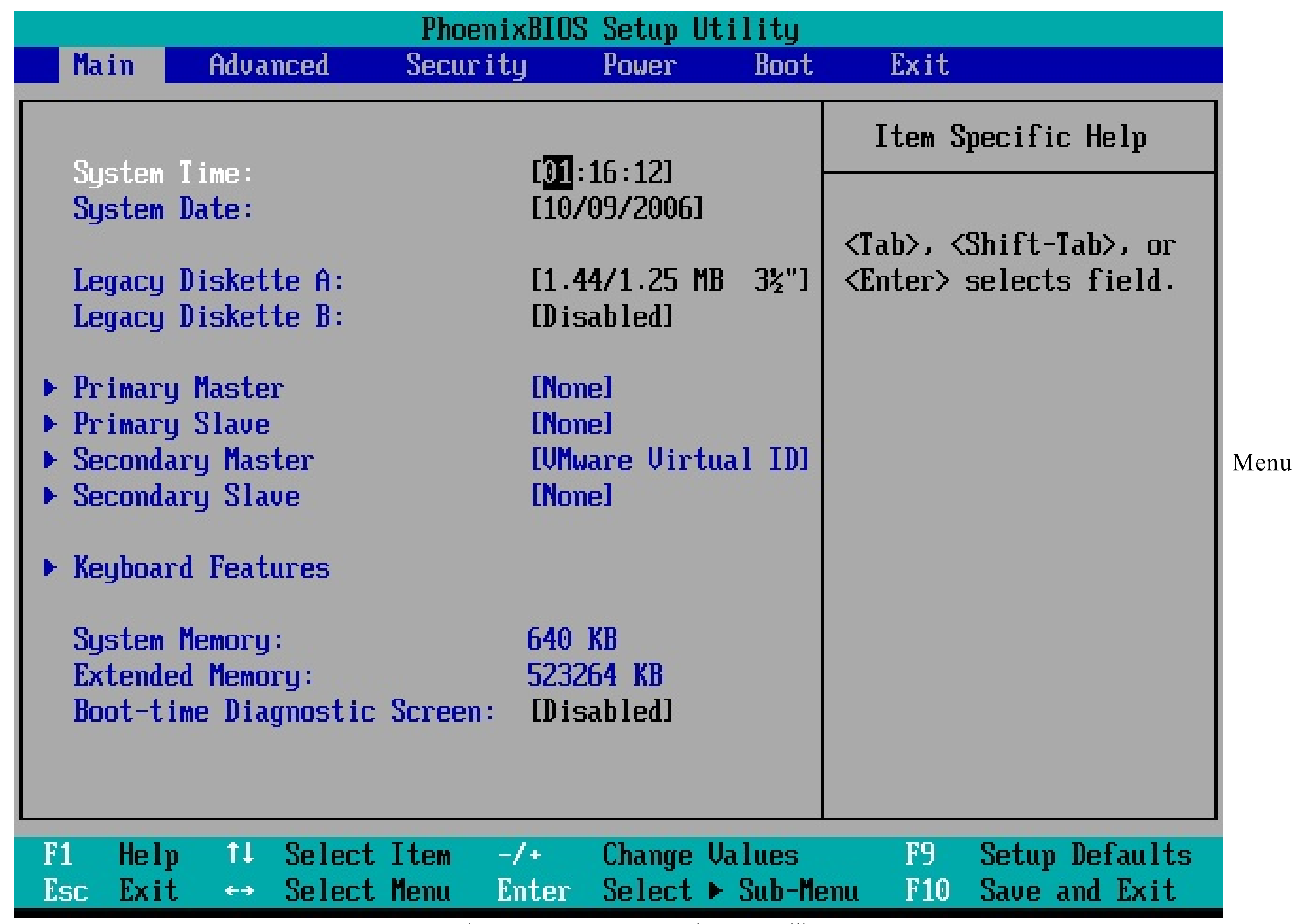
À cette étape, la plupart des PC démarrent sur le CD s'ils en trouvent un dans le lecteur, mais certains ordinateurs doivent être configurés pour démarrer à partir du CD. Pas de chance pour vous.
Heureusement, je vais vous expliquer comment faire pour changer cela.


Si vous n'avez pas eu de problème et que vous avez vu l'écran d'accueil d'Ubuntu dès le début, vous pouvez directement sauter à l'étape suivante.

Modifier l'ordre de boot

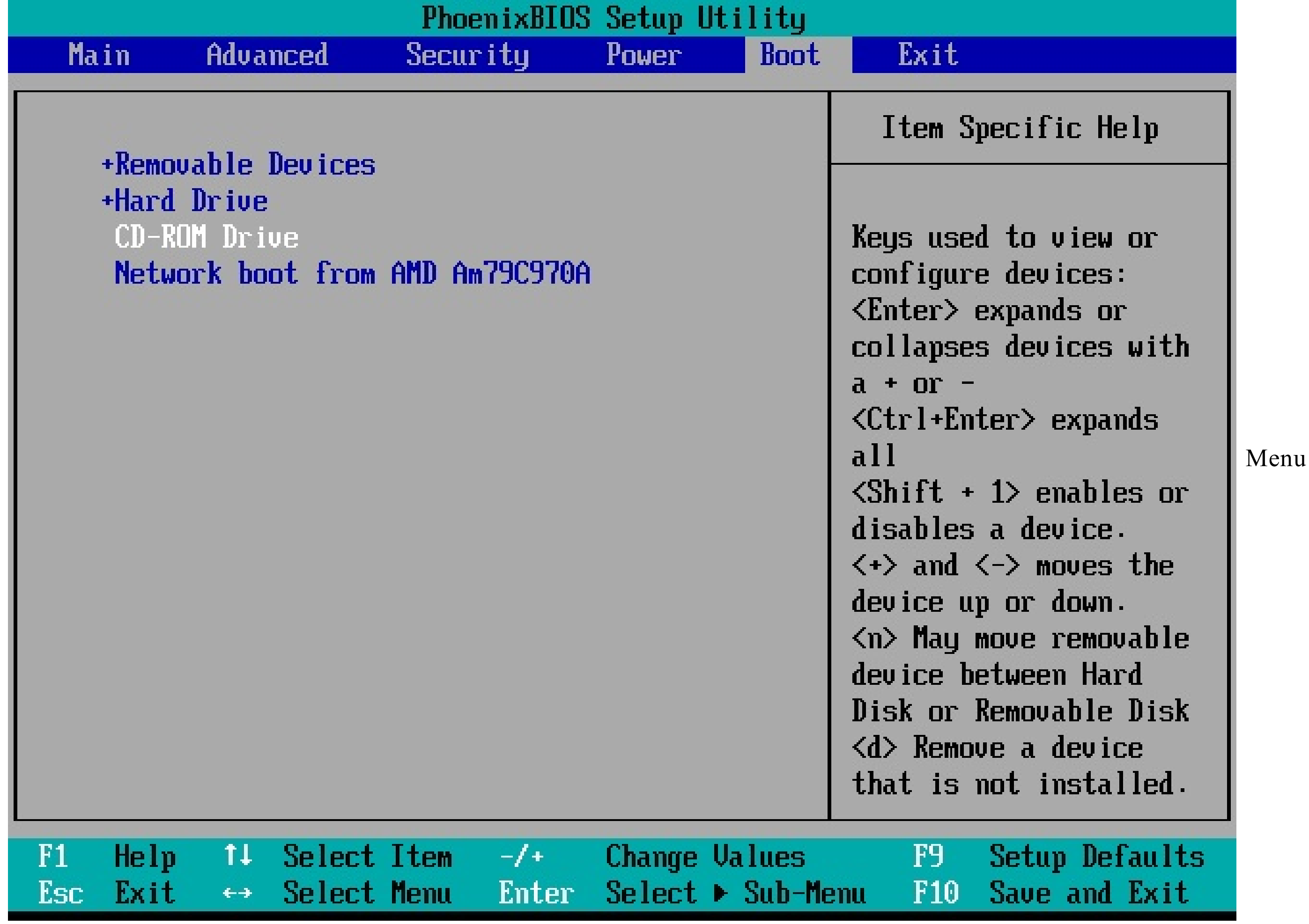
Si vous devez modifier l'ordre de boot pour que votre ordinateur lise le CD, redémarrez. Pendant l'écran de boot (la toute première chose que vous voyez à l'écran), pressez la touche indiquée pour accéder au *Setup*, aussi appelé **BIOS** (c'est l'écran de configuration de votre carte mère).
Généralement, la touche est `F1`, `F2` ou `Suppr`, mais cela peut varier selon le modèle de votre carte mère.

Vous devriez alors voir le superbe menu du BIOS (sigh !), comme sur la figure suivante. D'un ordinateur à l'autre, cet écran peut être légèrement différent.



Menu du BIOS : on a connu plus accueillant

Repérez le menu « Boot ». Il faudra généralement vous déplacer à l'aide des flèches du clavier. La figure suivante montre ce que vous devriez voir à peu de choses près (le menu « Boot » peut être différent sur votre ordinateur).



Menu

« Boot » dans le BIOS

Ici, on peut définir l'ordre dans lequel l'ordinateur essaie de démarrer les éléments. À vous de changer cet ordre pour faire en sorte que votre ordinateur essaie de démarrer sur le CD **avant** de démarrer sur le disque dur.

Lisez les instructions sur le côté (certes, en anglais), pour savoir comment faire sur votre ordinateur.
Si vraiment vous êtes bloqués, n'hésitez pas à aller demander de l'aide sur les forums du Site du Zéro.

Essayer ou installer Ubuntu

Bien ! À partir de maintenant, je suppose qu'Ubuntu se lance au démarrage de l'ordinateur. Après quelques instants de chargement, vous devriez voir l'écran présent sur la figure suivante.

Installation

Bienvenue

No localization
Asturianu
Bahasa Indonesia
Bosanski
Català
Čeština
Cymraeg
Dansk
Deutsch
Eesti
English
Español
Esperanto
Euskara
Français
Gaeilge
Galego
Hrvatski
Italiano
Kurdî

Essayer Ubuntu

Installer Ubuntu

Vous pouvez essayer Ubuntu sans rien changer à votre ordinateur, directement depuis ce CD.

Ou si vous êtes prêt, vous pouvez installer Ubuntu à côté (ou à la place) de votre système d'exploitation actuel. Cela ne devrait pas prendre beaucoup de temps.

Premier accueil d'Ubuntu

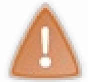
Sélectionnez la langue dans le menu de gauche si les textes ne sont pas en français.

Vous voyez que vous avez deux choix :

- **Essayer Ubuntu** : Ubuntu sera lancé sans toucher à votre disque dur, vous pourrez donc l'essayer pour le tester ;
- **Installer Ubuntu** : Ubuntu sera installé sur votre disque dur. Utilisez ce choix si vous êtes déjà certains de vouloir installer Ubuntu.

Par la suite, je vais supposer que vous avez fait le premier choix (celui que tous les débutants seraient tentés de faire !).

Ubuntu va alors se charger à partir du CD dans la mémoire vive. En effet, dans ce mode de test appelé « Live CD », Linux ne touche pas à votre disque dur. Aucun risque d'altérer Windows, tout est fait dans la mémoire vive (et entre nous, c'est une sacrée prouesse technique !).



En mode Live CD, tout est chargé depuis le CD… or, un CD est beaucoup plus lent qu'un disque dur !
Si le chargement de Linux s'éternise, ne paniquez pas, c'est tout à fait normal : la lecture depuis le CD prend **beaucoup de temps**. Rassurez-vous : une fois Linux installé sur le disque dur, le chargement sera bien plus rapide que cela. ;-)

Une fois le chargement effectué, vous allez rapidement vous retrouver sur le gestionnaire de bureau que vous avez choisi. Si, comme moi, vous avez opté pour Ubuntu, vous serez donc sous Unity.

Le bureau de Unity ressemble à la figure suivante (il peut légèrement varier en fonction de votre version d'Ubuntu).

Ubuntu : le bureau Unity, une fois démarré

Alors, qu'en dites-vous ? Ce n'était pas franchement la mer à boire !

Retenez bien : tout ce que vous voyez là a été chargé dans votre mémoire vive. Ubuntu n'a pas touché à votre disque dur. Vous pouvez donc tester Linux en toute sécurité. N'hésitez pas à parcourir les menus et à vous familiariser un peu avec l'environnement de bureau que vous avez choisi.



Je détaillerai le fonctionnement des gestionnaires de bureau **KDE** et **Unity** dans les prochains chapitres.
Pour le moment, je vous laisse le soin de découvrir un peu par vous-mêmes, j'estime que c'est important.

Installer Ubuntu

Alors, vous avez fait un petit tour dans les programmes fournis avec Ubuntu ?

Sachez que selon la version que vous avez prise (Ubuntu, Kubuntu ou Xubuntu), les programmes installés par défaut seront différents.
En effet, certains programmes sont à la base destinés à KDE, mais fonctionnent aussi sans problème sur Unity. Inversement : certains programmes sont destinés, à la base, à Gnome, mais on peut très bien les utiliser sous Unity et KDE (c'est le cas de Firefox, par exemple). Il n'y a donc pas d'incompatibilité entre les gestionnaires de bureau, mais on préfère généralement installer au départ les programmes prévus pour KDE sur KDE, histoire d'être… logique.

Allez, je vous sens chauds pour une petite installation, là.
On y va ?



Attention : même si l'installation est très détaillée et sécurisée, il y a toujours un petit « risque » que vous installiez par-dessus Windows. Dans tous les cas, **faites une sauvegarde de vos fichiers les plus importants** avant de commencer l'installation, sur CD ou clé USB par exemple.
Pas de panique, tout va bien se passer, mais en général deux protections valent mieux qu'une.

Étape 1 : lancer l'installation et choisir la langue

Vous allez voir : contrairement à ce que vous pensiez il y a quelques minutes à peine, installer Linux est d'une simplicité… frustrante.
Il faut d'abord ouvrir le programme d'installation qui se trouve sur le bureau en cliquant sur l'icône de la figure suivante.



Icône d'installation d'Ubuntu


Mes captures d'écran sont faites sur la version 12.04 d'Ubuntu. Il est possible que vous ayez téléchargé une version plus récente (il en sort tous les 6 mois !), mais rassurez-vous : malgré quelques légères différences, le principe de l'installation reste toujours le même.

La première fenêtre de l'assistant s'ouvre (figure suivante). Vérifiez que « Français » est bien sélectionné.

Installation

Bienvenue

Deutsch
Eesti
English
Español
Esperanto
Euskara
Français
Gaeilge
Galego
Hrvatski
Italiano
Kurdî
Latviski

Vous pouvez éventuellement lire les notes de publication.

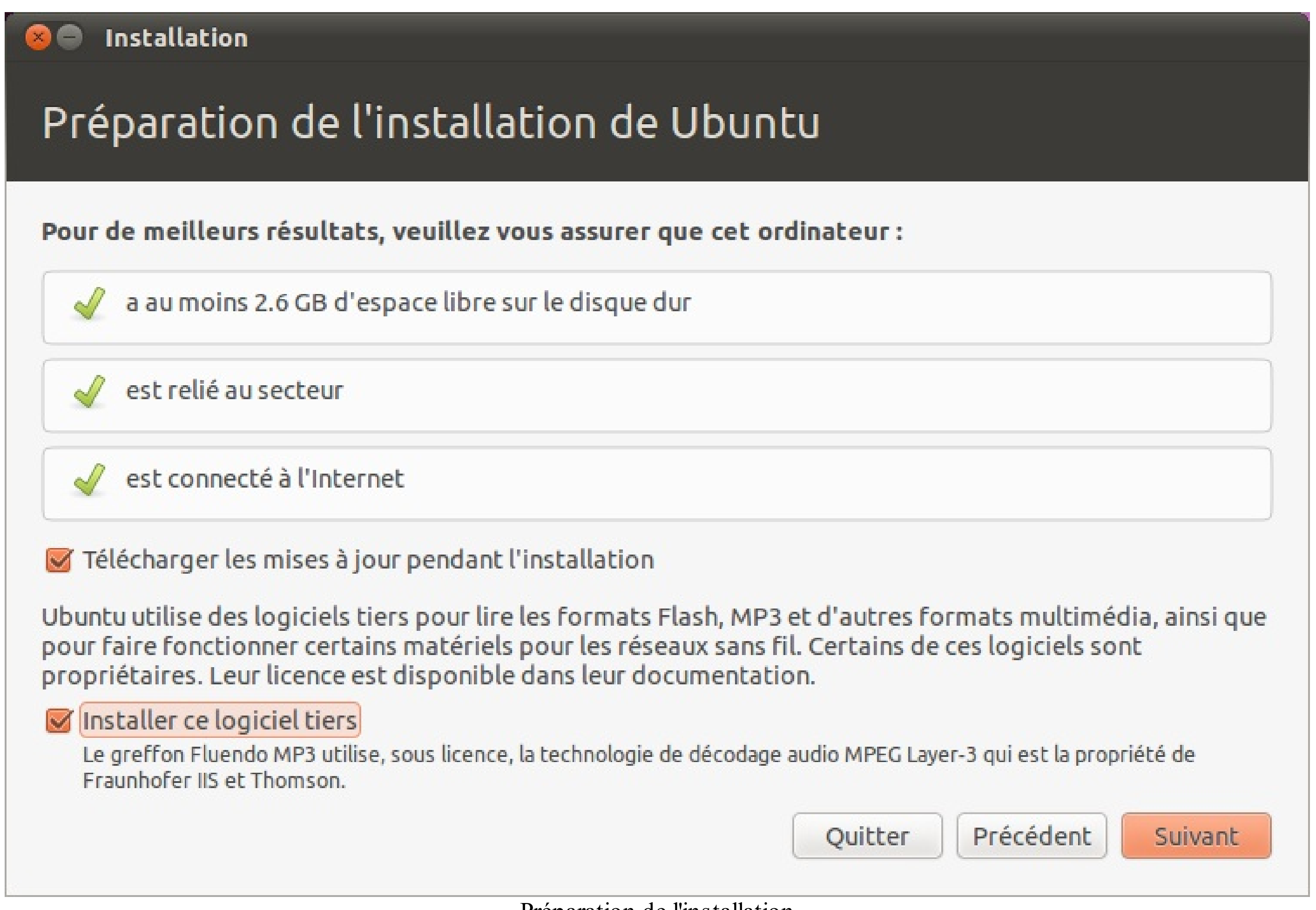
Quitter
Précédent
Suivant

Choix de la langue

Cliquez ensuite sur « Suivant » pour passer à l'étape n° 2.

Etape 2 : préparation de l'installation

On vous demande de vous préparer à l'installation d'Ubuntu sur votre ordinateur :



Préparation de l'installation

3 conditions sont posées :

- **Avoir un minimum d'espace disque disponible** : normal, si vous voulez avoir la place d'installer Ubuntu ! Ici, il s'agit du strict minimum, je vous conseille d'avoir 8-10 Go pour être suffisamment confortable. A priori, pas besoin de plus d'espace supplémentaire, sauf si vous prévoyez d'y stocker de gros fichiers : musique, photos, vidéos perso etc.
- **Etre branché sur le secteur** : cela concerne bien entendu les ordinateurs portables. Il est très fortement recommandé d'être branché, car installer un système d'exploitation sur batterie est tout simplement... suicidaire. Il serait très ennuyeux pour votre installation que celle-ci soit coupée en plein milieu à cause d'une batterie vide !
- **Etre connecté à Internet** : c'est facultatif, mais je le recommande fortement là aussi. Cela permettra à l'assistant d'installation de télécharger immédiatement les dernières mises à jour des programmes ainsi que les traductions françaises qui pourraient manquer sur le CD d'Ubuntu. Oubliez le wifi ici, qui peut être un peu compliqué à configurer : branchez-vous à Internet avec un vrai câble réseau (RJ45).

Deux options peuvent être cochées (et je recommande de les cocher toutes les deux !) :

- **Télécharger les mises à jour pendant l'installation** : cela vous assurera que les programmes sont immédiatement le plus à jour possible. C'est préférable car les mises à jour corrigent des failles de sécurité, des bugs et améliorent certaines fonctionnalités des programmes déjà présents sur votre CD d'Ubuntu. Bien entendu, il faut être connecté à Internet avec un câble réseau pour cela.
- **Installer ce logiciel tiers** : cette option vous permet d'installer certains programmes propriétaires. Pour qu'Ubuntu reste libre, ces programmes ne sont pas installés par défaut, mais vous pouvez *demander* leur installation en cochant cette case (ce que je vous recommande pour votre confort). Vous aurez ainsi la possibilité de lire des MP3, du Flash et d'autres fichiers multimédia protégés par des licences propriétaires. Cette option peut aussi améliorer la prise en charge de votre carte wifi. Bref, c'est forcément intéressant pour vous, sauf si vous ne voulez pas installer de programme propriétaire sur votre machine.

Etape 3 : partitionnement du disque dur

Nous arrivons maintenant à une étape importante de l'installation : le partitionnement du disque dur. C'est un sujet intéressant, tellement intéressant que je vais devoir y dédier l'intégralité du prochain chapitre.
Vous aurez donc fini d'installer Linux à l'issue de celui-ci !

En résumé

- Il est possible d'installer Ubuntu directement depuis Windows, mais la manipulation classique consiste plutôt à redémarrer l'ordinateur avec le CD d'Ubuntu dans le lecteur.
- La première fois, Ubuntu se charge directement sur le CD, en mode *Live CD*. Votre disque dur n'est pas modifié et vous pouvez tester Ubuntu tranquillement.
- Si vous êtes décidés, il suffit de lancer le programme d'installation depuis Ubuntu et de suivre les étapes.

 Partitionner son disque

L'étape du partitionnement est probablement l'une des plus importantes de l'installation de Linux. Elle consiste à découper votre disque en plusieurs parties afin, par exemple, de ne pas mélanger Linux et Windows (ça ferait désordre).

Avant de partitionner votre disque, nous allons découvrir ce qu'est le partitionnement et à quoi cela sert. Ensuite, nous verrons comment effectuer le partitionnement à l'aide de l'outil proposé lors de l'installation d'Ubuntu. Soyez attentifs, car il faut que cette étape soit réalisée au mieux. Il est en effet délicat de faire marche arrière une fois le partitionnement effectué.



Bien que les risques soient minimes si vous suivez pas à pas mes instructions, je vous recommande quand même de **faire une sauvegarde de vos données importantes sous Windows** avant de commencer ce chapitre. Le partitionnement, si vous l'effectuez mal, pourrait affecter la partition Windows. Pas de panique, tout sera expliqué, mais on n'est jamais trop prudent non plus.

Défragmentez votre disque

Dans ce chapitre, on va beaucoup parler de votre disque dur. En fait, on ne va parler que de lui. C'est son organisation qui nous intéresse.

Un disque dur, ça ressemble à quoi ?

Pour bien commencer à partir de Zéro, je pense qu'il serait bien que je montre à quoi ressemble un disque dur à ceux qui n'en ont jamais vu de leur vie. La figure suivante montre un disque dur tel qu'on peut en trouver sur le marché ; vous en avez forcément un dans votre ordinateur.



Un disque dur, vu de l'extérieur

Un disque dur ne doit **jamais** être ouvert, vous risqueriez d'endommager vos données. Cependant, et pour des raisons purement pédagogiques, on va en ouvrir un pour bien comprendre ce qu'il y a à l'intérieur (figure suivante).



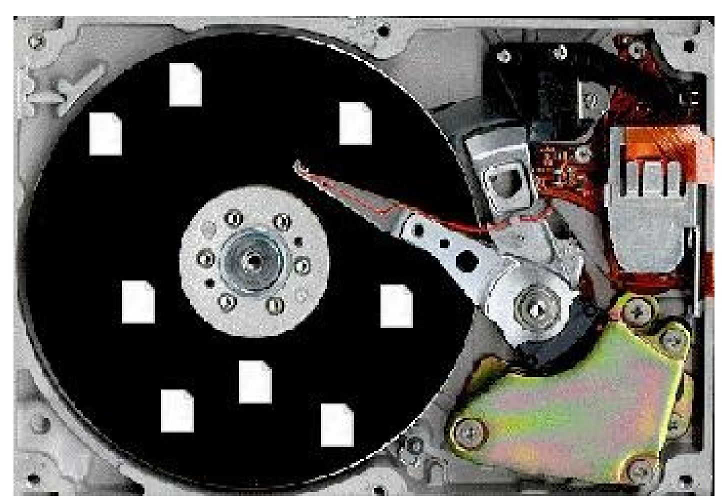
L'intérieur d'un disque dur

Comme vous le voyez, un disque dur est un empilement de disques, un peu comme des CD. Ils sont lus par une tête de lecture qui n'est pas sans rappeler la tête de lecture des disques vinyle.

L'importance de la défragmentation

Avant d'aller plus loin, il est très vivement conseillé d'effectuer une **défragmentation**. C'est une opération qui consiste en gros à mieux organiser les fichiers sur votre disque dur, à les rassembler pour éviter qu'ils ne soient éparpillés.

On ne dirait pas comme ça, mais vos fichiers sont parfois placés un peu n'importe comment à la surface de votre disque dur ! Voici un petit schéma sur la figure suivante pour bien comprendre dans quel état est votre disque.



Avant la défragmentation, c'est le bazar sur votre disque

Sur la surface du disque, j'ai représenté une multitude de fichiers : ce sont les fichiers tels qu'ils sont placés sur votre disque actuellement. Un beau bazar. Parfois, certains sont coupés en plusieurs morceaux et dispersés sur votre disque ! On dit que les fichiers sont **fragmentés** (coupés en plusieurs fragments).

Comment résoudre cela ? Votre ordinateur sait le faire, mais ça demande du temps. Grâce à un outil intelligent, appelé *défragmenteur*, il peut partir à la recherche des fichiers fragmentés et les rassembler tous au même endroit, comme sur la figure suivante.


Après la défragmentation, c'est bien plus propre !

Les avantages seront les suivants.

- Comme vos fichiers seront près les uns des autres, le disque dur mettra moins de temps à les récupérer quand on les lui demandera. Finalement, votre Windows sera sensiblement plus rapide (surtout si vous n'avez jamais défragmenté et que vous utilisez votre ordinateur depuis des années !).
- Et surtout, c'est ce qui nous intéresse ici, cette organisation « prépare » votre disque au partitionnement que vous allez faire. Si vous ne le faites pas, il y a un risque (j'ai bien dit un « risque ») que certains bouts de fichiers disparaissent lors du partitionnement : vous pourriez alors vous retrouver avec un Windows instable !

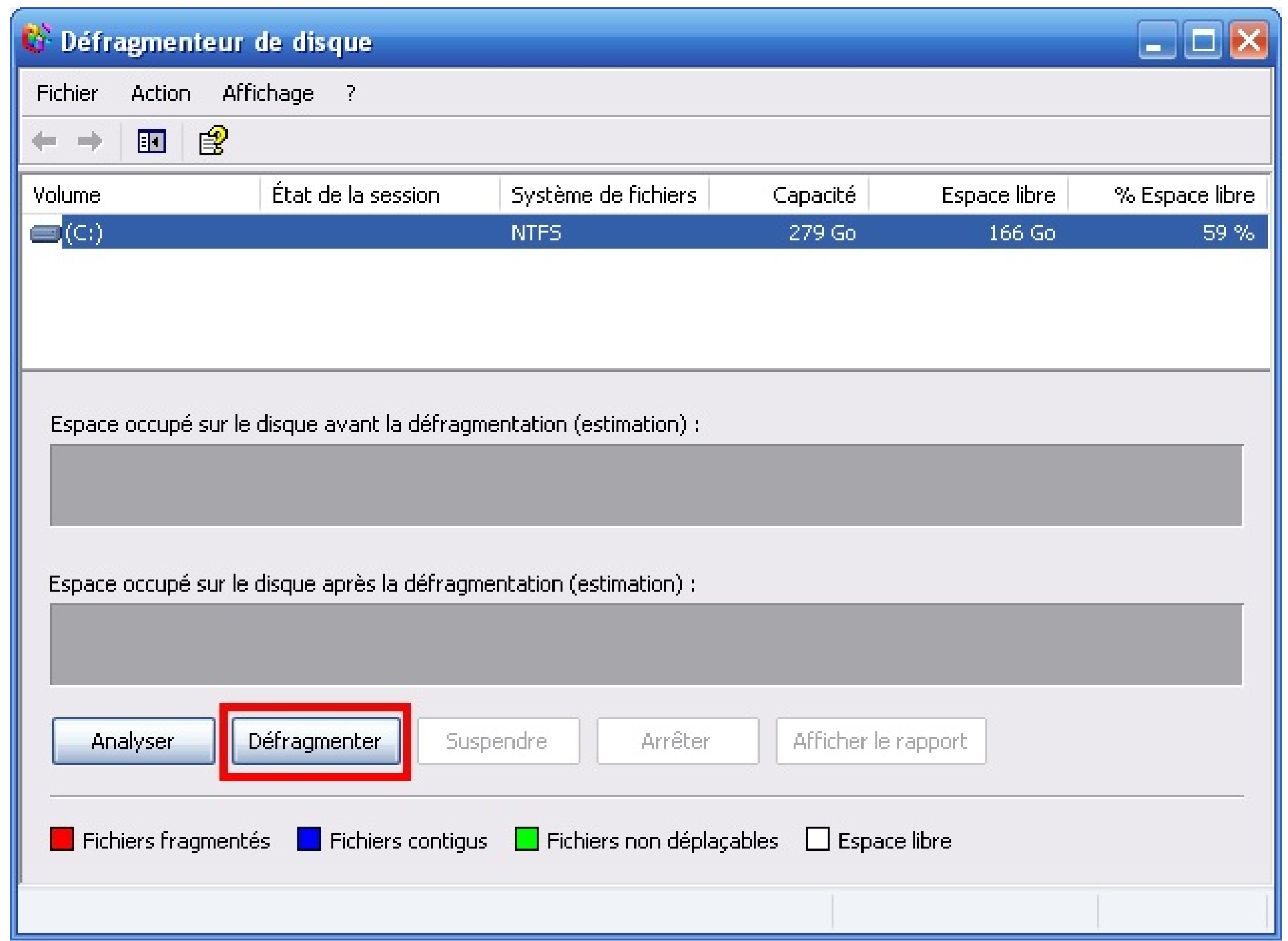
Bon, vous avez compris, il ne faut pas chercher à discuter : défragmentez votre disque, vous n'en tirerez que des avantages. :)

Pour défragmenter, un utilitaire est livré avec Windows. Retournez donc sous Windows pour effectuer la défragmentation si vous ne l'avez pas faite auparavant, c'est vraiment une étape importante.

Pour lancer cet utilitaire, allez dans : `Démarrer` → `Tous les programmes` → `Accessoires` → `Outils Système` → `Défragmenteur de disque`.

Sous Windows XP

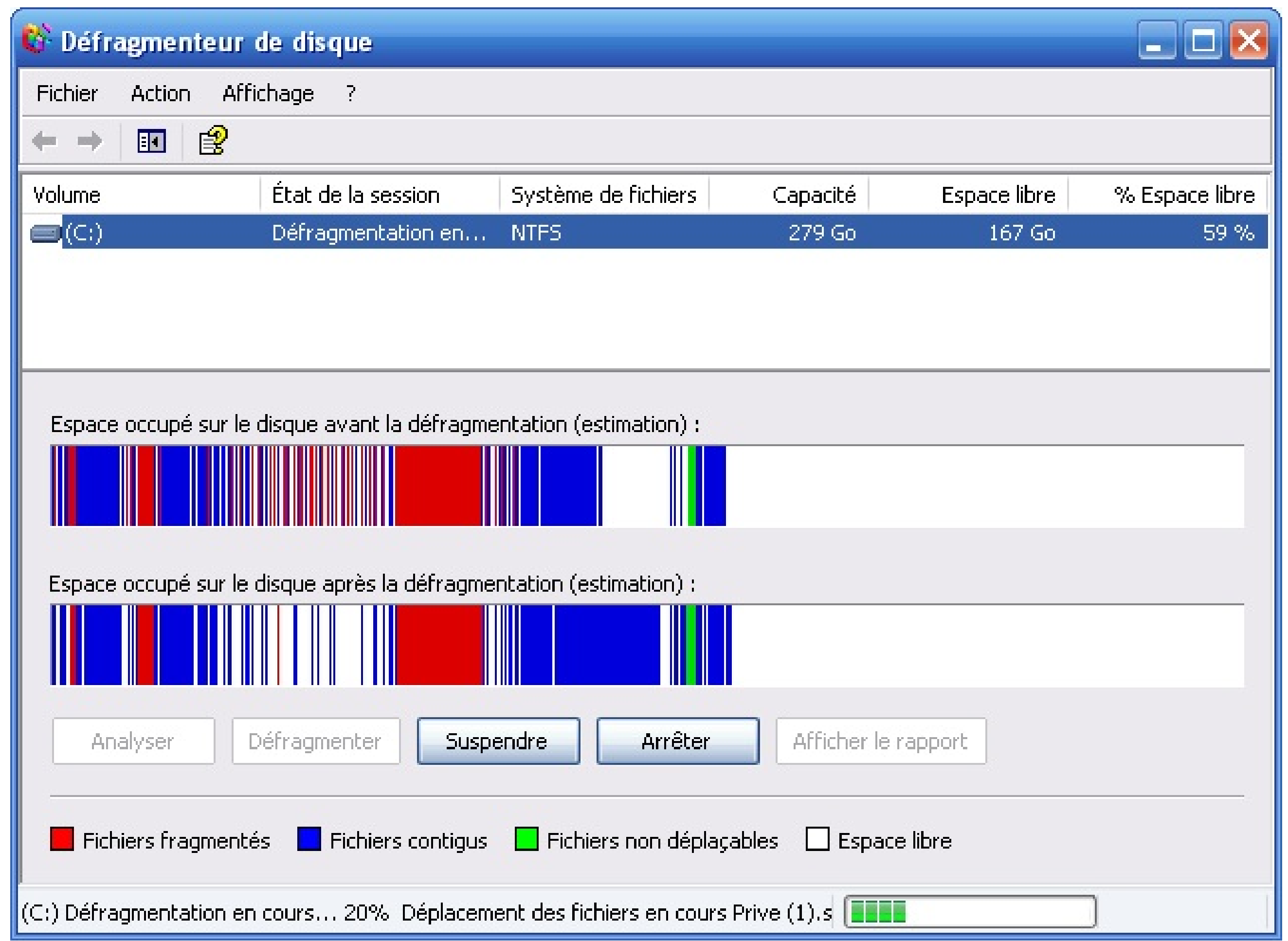
La fenêtre de la figure suivante s'ouvre.



Défragmenteur de disque

Sélectionnez le disque dur que vous voulez défragmenter (celui qui contient Windows, généralement `C:`) et cliquez sur le bouton « Défragmenter ».
Vous pouvez sortir prendre l'air, parce que la défragmentation peut prendre un moment (plusieurs heures si vous n'en avez jamais fait).

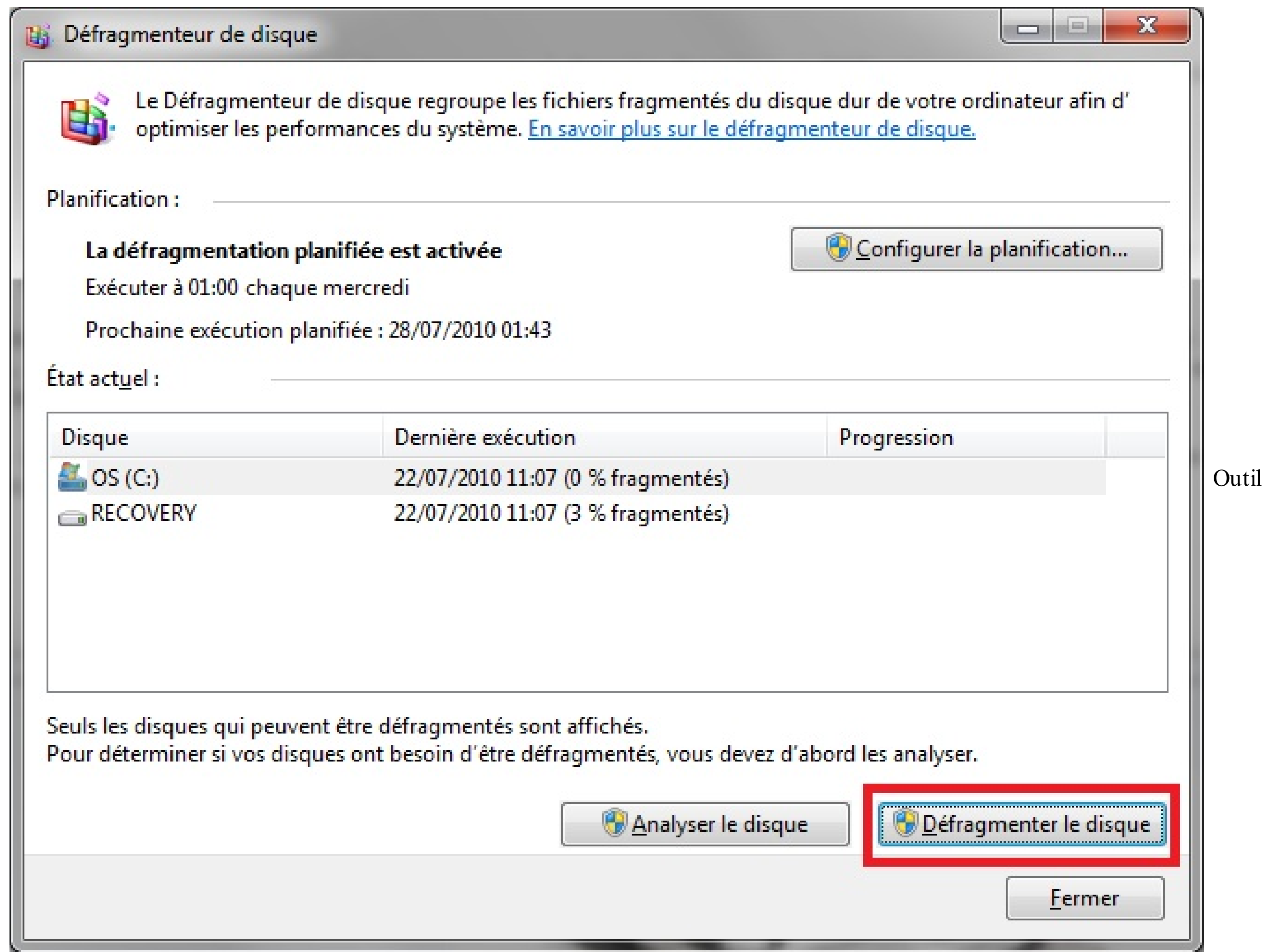
Pendant la défragmentation, l'avancement est indiqué en bas de la fenêtre (figure suivante).



Défragmentation en cours

Sous Windows 7

La fenêtre est sensiblement différente (figure suivante), mais le principe est le même.



Outil de défragmentation de Windows 7

Il suffit de sélectionner le disque à défragmenter (si vous en avez plusieurs). Il est conseillé de tous les défragmenter. Cliquez ensuite sur « Défragmenter le disque ».

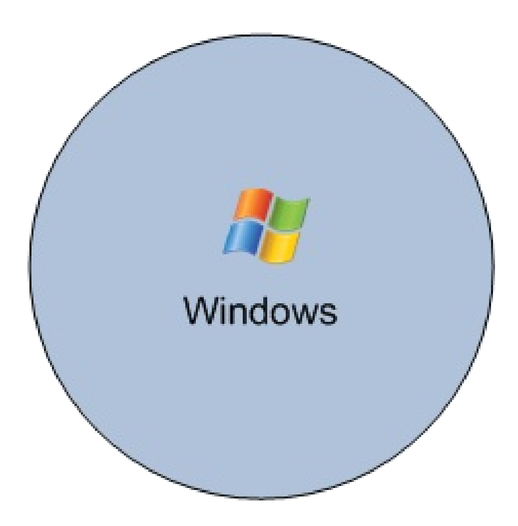
N'éteignez pas votre PC pendant la procédure et attendez sagement. Idéalement, évitez de trop toucher à votre ordinateur pendant la défragmentation pour ne rien perturber (ce n'est pas interdit, mais c'est déconseillé).

Une fois que c'est terminé, vous aurez un disque dur tout propre et bien organisé.

Qu'est-ce que le partitionnement ?

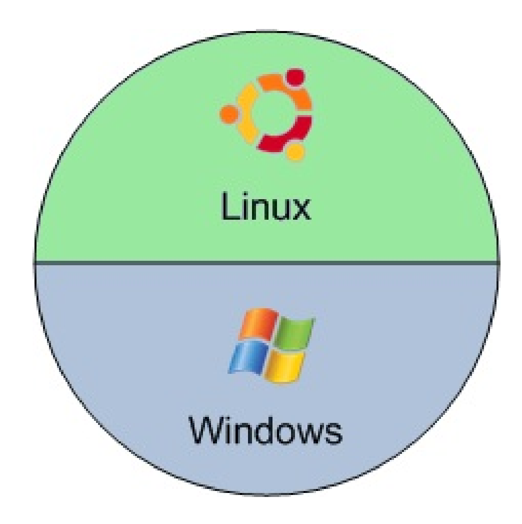
Vive les schémas !

Pour expliquer le principe du partitionnement du disque dur, je vais avoir recours à quelques schémas.
Je vais représenter votre disque dur par un cercle. Et qu'y a-t-il dessus, actuellement ? Il y a de fortes chances pour qu'il n'y ait que Windows ! Votre disque dur ressemblerait donc à la figure suivante.

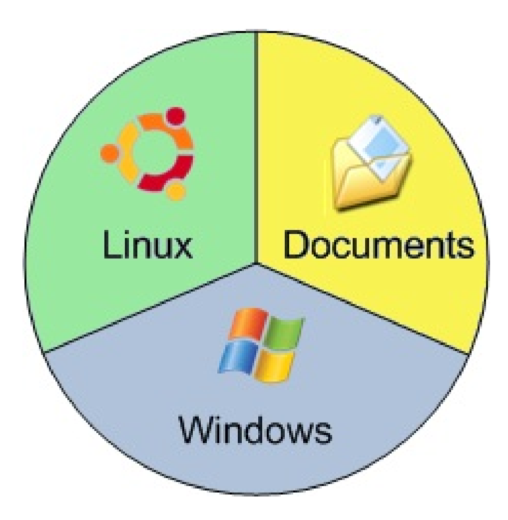


Votre disque « appartient » grosso modo à Windows. Il s'est installé dessus et il considère que tout lui appartient, donc qu'il peut mettre des fichiers où il veut sur le disque.

Supposons maintenant que l'on introduise Linux. Il est impossible de mettre 2 OS (ici, Windows et Linux) ensemble au même endroit. Cela engendrerait trop de conflits. Pour résoudre le problème, on a inventé le partitionnement. Cela consiste à découper son disque en plusieurs parties (virtuellement, hein, pas pour de vrai).
Si on allouait 50 % de l'espace à Linux et 50 % à Windows, le schéma ressemblerait alors à la figure suivante.



Mieux encore, le fin du fin serait de créer une partition où l'on placerait nos documents, comme sur la figure suivante.



Ainsi, Linux et Windows pourraient s'échanger vos fichiers (photos, documents texte, etc.) sans interférer l'un avec l'autre. Autre intérêt de cette technique : si par hasard vous deviez réinstaller Linux ou Windows, vous ne perdriez pas vos documents car seules les partitions Linux ou Windows seraient formatées !

En quelque sorte, mettre vos documents dans une partition séparée, c'est les mettre à l'abri.

Les systèmes de fichiers

Sur chaque partition, les fichiers sont organisés selon ce qu'on appelle un **système de fichiers**. C'est en quelque sorte une façon d'organiser les fichiers : ils sont tous référencés dans une sorte d'annuaire gigantesque.
Le système de fichiers permet aussi de dire qui a le droit de voir tel ou tel fichier. D'autre part, les systèmes de fichiers récents sont dits « journalisés », c'est-à-dire qu'en cas de crash (votre PC est éteint brusquement), le système est capable de retrouver ses fichiers sans trop de problèmes.

Règle importante : il ne peut y avoir qu'**un seul système de fichiers par partition**.

Il existe un nombre important de systèmes de fichiers différents, en voici quelques-uns à connaître.

Systèmes de fichiers Microsoft (DOS et Windows)

- **FAT 16** : un très vieux système de fichiers, capable de gérer jusqu'à 4 Go de données. Il est donc impossible de faire une partition en FAT 16 de plus de 4 Go. Il était très utilisé à l'époque du DOS et aux débuts de Windows 95.
- **FAT 32** : une évolution du FAT 16, qui pousse la limite de taille à 2 To (2 000 Go). Le FAT 16 et le FAT 32 ont la particularité de beaucoup fragmenter les fichiers, d'où la nécessité de défragmenter régulièrement, sinon on prend le risque de voir son disque ressembler à un véritable champ de bataille.
- **NTFS** : apparu avec Windows NT, puis réutilisé par Windows XP, il permet de créer des partitions d'une taille allant jusqu'à 16 Eo (16 *Exaoctets*, soit 16 000 000 000 de Gigaoctets). Mais ce n'est pas tant la taille maximale qui est intéressante (on en est franchement loin) que les avantages que le NTFS procure à côté. Contrairement au FAT 32, c'est un système de fichiers journalisé qui récupère beaucoup mieux les données en cas de crash du disque. D'autre part, on peut donner des droits sur certains fichiers, les crypter, les compresser, etc. Enfin, les fichiers sont censés moins se fragmenter… cependant, ils se fragmentent toujours.

Systèmes de fichiers Linux

- **ext2** : c'est le système de fichiers qui a longtemps été utilisé sous Linux. Il a été développé par un français (Rémy Card) et présente la particularité de très peu se fragmenter. Ainsi, sous Linux et depuis longtemps, il n'y a pas besoin de faire de défragmentation.
- **ext3** : l'ext3 est très proche de l'ext2, à une différence majeure près : la journalisation. En effet, ext2 n'était pas journalisé et en cas de crash du disque, on risquait plus facilement une perte de données. Ce n'est plus le cas avec l'ext3.
À noter que l'ext2 et l'ext3 sont parfaitement compatibles entre eux, dans un sens comme dans l'autre.
- **ext4** : une amélioration de l'ext3, relativement récente, qui améliore la prise en charge des gros disques durs et diminue les problèmes de fragmentation des fichiers.

Lequel choisir ?

Actuellement, et pour faire simple, la plupart des ordinateurs sous Windows utilisent le NTFS et sous Linux, la plupart utilisent l'ext3.
Nous allons choisir l'ext4, plus récent et plus performant.

Je peux vous proposer ceci pour vos partitions :

- **Linux** : ext4 ;
- **Windows** : NTFS (ça ne bouge pas) ;
- **Documents** : ext4.

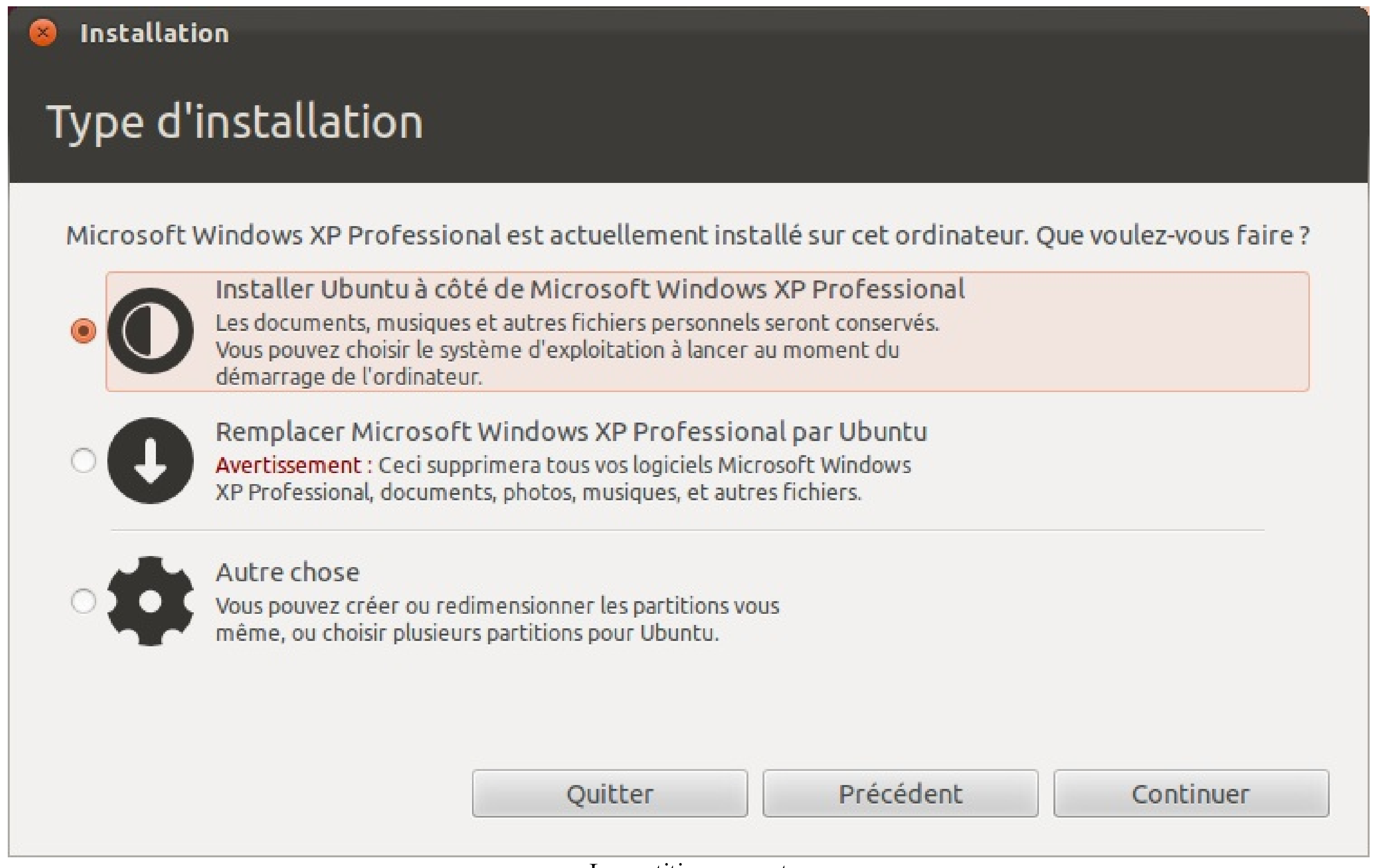
Le but est que la partition « Documents » puisse être lue et écrite depuis Windows et Linux, afin que vous puissiez accéder à vos documents, que vous soyez sous Windows ou Linux.

L'ext4, comme vous le savez, fonctionne parfaitement sous Linux. Cependant, Windows ne le reconnaît pas… à moins d'installer un programme qui lui permettra de l'utiliser. Je peux vous recommander **FS Driver**, qui rend utilisables l'ext2, l'ext3 et l'ext4 sous Windows.

Prêts ? Partitionnez !

Revenons à l'installation d'Ubuntu, et plus précisément à l'étape 4, celle du partitionnement.

La fenêtre de la figure suivante vous propose plusieurs options dans le cas où Windows est déjà installé.



Le partitionnement

- **Installer les deux côte à côte** : Ubuntu va se faire automatiquement de la place sur votre disque dur et créer les partitions pour vous. C'est la solution la plus simple que vous devriez choisir si vous ne voulez pas entrer dans les détails. En revanche, vous n'aurez pas de partition spéciale pour les documents dans ce mode-ci.
  En bas de la fenêtre, vous pouvez déplacer le curseur pour décider de l'espace que vous attribuez à Windows et à Ubuntu.
- **Tout effacer et utiliser le disque entier** : tout le disque sera formaté, partition Windows comprise. Ne faites cela que si vous voulez supprimer Windows ! Ubuntu sera installé sur l'ensemble du disque dur.
- **Définir les partitions manuellement (avancé)** : choisissez cette option si vous voulez créer vous-mêmes les partitions. C'est plus complexe mais cela vous donnera plus de choix.

En résumé : si vous voulez aller vite et faire simple, choisissez l'option « Installer les deux côte à côte ». Je vous le conseille si vous n'avez pas envie d'y passer trop de temps.
Sinon, choisissez le mode manuel : nous allons découvrir ci-dessous comment celui-ci fonctionne.

Le partitionnement manuel

Je vais supposer que vous procédez à un partitionnement manuel, et donc vous montrer pas à pas comment ça fonctionne avec l'outil de partitionnement présent dans le gestionnaire d'installation d'Ubuntu. Mais avant cela, il faut que je vous fasse découvrir comment sont nommées les partitions du disque !

À propos du nom des disques

Si vous avez un seul disque dur sur votre ordinateur, pas de problème, vous ne risquez pas de vous tromper. Toutefois, si par hasard vous avez plusieurs disques, je pense que vous apprécierez que je vous explique comment les disques durs sont nommés sous Linux.

En effet, c'est très différent de Windows où l'on a l'habitude des sempiternels `C:`, `D:`, `E:`, etc.
On va découvrir les noms des disques sous Linux avec un exemple : **hda**.

- **h** : la première lettre indique si le disque est de type IDE ou SCSI (un type de connexion différent à la carte mère). S'il est de type IDE, la lettre est un « h », si c'est un SCSI (ou un S-ATA), la lettre est un « s ».
- **d** : cette lettre ne change pas.

- **a** : c'est cette lettre qui indique les différents disques durs. *hda* représente le premier disque dur IDE, *hdb* le second, *hdc* le troisième, etc.

Lorsque l'on crée des partitions, on ajoute généralement un chiffre représentant le numéro de la partition. Ainsi, si on a trois partitions sur notre disque *hda*, elles seront nommées *hda1*, *hda2*, *hda3*…

L'outil de partitionnement manuel

Revenons à notre installation d'Ubuntu. Si vous avez choisi le partitionnement manuel, vous devriez voir l'écran de la figure suivante.

Le partitionnement manuel

Sur la première ligne, vous avez le nom du disque dur : `/dev/sda`. C'est donc le disque dur *sda*. Comme je n'ai qu'un seul disque dur, il est facile à reconnaître.
En dessous, on voit les partitions que contient le disque. Ici, il y en a une seule (qui correspond à Windows).

Actuellement, j'ai une partition nommée *sda1* car, comme je vous l'ai expliqué un peu plus haut, les partitions sont numérotées. Quand on va rajouter des partitions, vous allez voir qu'elles vont s'appeler *sda2*, *sda3*, etc.
Cette partition est de type NTFS, le système de fichiers de Windows. C'est donc la partition utilisée par Windows, et c'est là que Windows est installé.



Il se peut que vous ayez plus de partitions que moi. Par exemple, il est fréquent de voir certains PC achetés dans le commerce équipés d'une toute petite partition « restore », capable de restaurer certains logiciels de votre ordinateur en cas de formatage. Si vous en avez une, n'y touchez pas. Votre PC est peut-être « tatoué ». Pour plus d'informations, lisez la documentation Ubuntu à ce sujet.
Si cela vous semble trop compliqué, vous pouvez toujours opter pour une installation simple depuis Windows qui ne pose pas ce genre de problème de partitionnement. Nous avons décrit cette procédure d'installation « simplifiée » au

début du chapitre précédent.

Étape 1 : réduire la taille de la partition Windows

La première étape consiste à limiter l'espace dédié à Windows pour faire de la place pour Ubuntu.

Sélectionnez la partition Windows en cliquant dessus (`/dev/sda1` de type NTFS dans mon cas), puis cliquez sur le bouton « Modifier ». Dans la fenêtre qui s'ouvre, vous pourrez modifier la taille de la partition en entrant une nouvelle taille, plus petite, dans le premier champ, puis en cliquant sur « Valider » (figure suivante).



Redimensionnement de la partition Windows

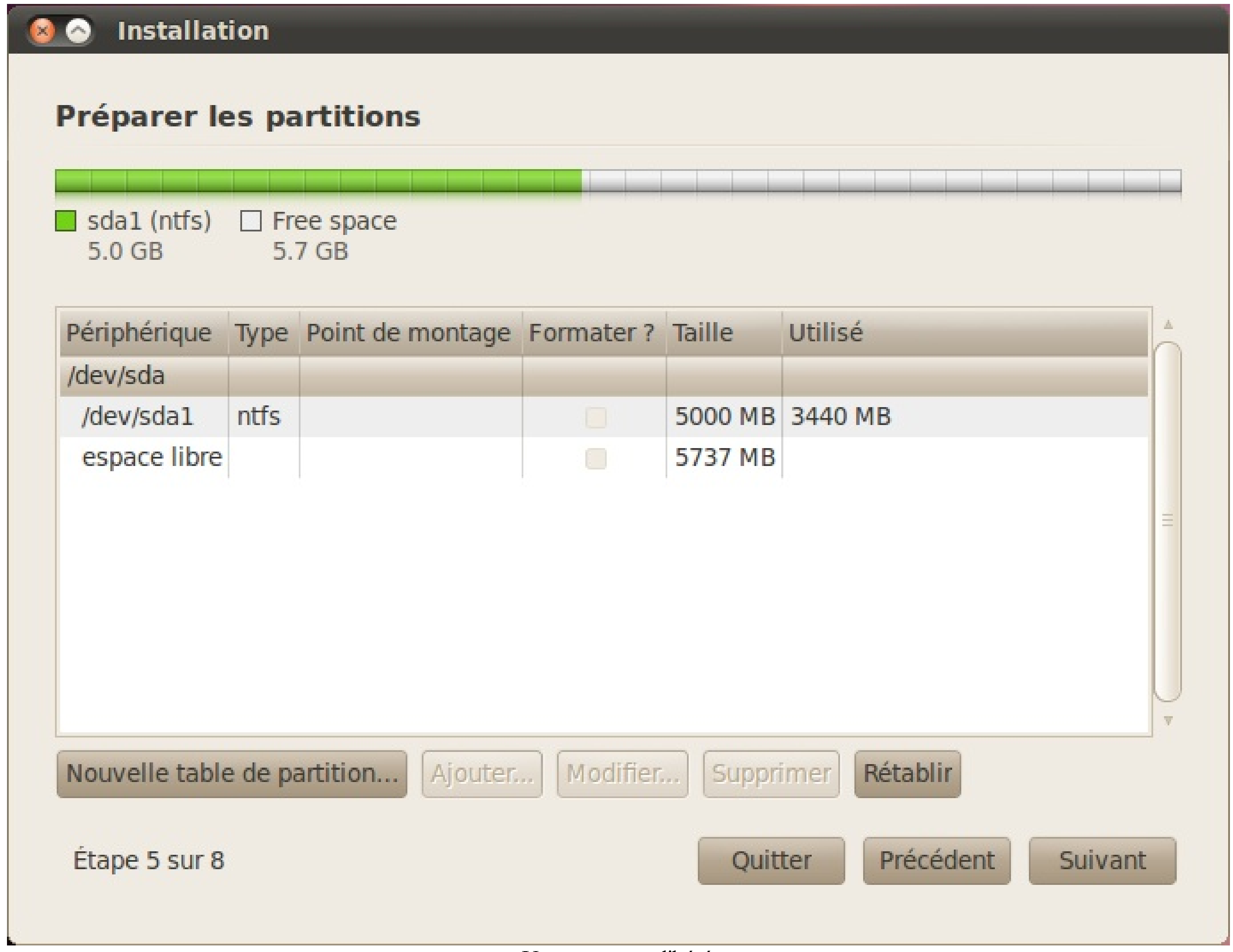
La taille est exprimée en Mo.
1 000 Mo font environ 1 Go. Indiquez la taille que vous souhaitez laisser à Windows.



Veillez à laisser tout de même assez de place à Windows, sinon vous ne pourrez plus faire grand-chose dessus (impossible d'installer un nouveau jeu s'il n'y a plus de place sur la partition, par exemple).

C'est à ce moment-là que la défragmentation se révèle utile. Comme toutes les données ont été groupées au même endroit, ça évite que certains fichiers égarés soient accidentellement supprimés. Ce serait un tantinet ballot, avouez.

Vous devriez alors avoir de l'espace libre, comme sur la figure suivante.



Un espace est libéré

Étape 2 : créer une partition pour installer Ubuntu

Ubuntu vous propose de créer deux types de partitions :

- **primaire** : c'est la partition de base, classique, on ne peut en créer que quatre par disque ;
- **logique** : c'est un type de partition qui peut contenir de nombreuses sous-partitions. Celle-ci n'est pas limitée en nombre, à la différence de la partition primaire.

Cliquez sur la partition libre du disque dur, puis sur le bouton « Nouvelle table de partition… » en bas.



Création de la partition Ubuntu

Dans la fenêtre qui s'ouvre (figure suivante), je vous invite à créer une partition « Primaire », de la taille que vous voulez, qui servira à installer Ubuntu ainsi que de futurs programmes. Indiquez au moins 3 ou 4 Go.
Laissez « Emplacement de la nouvelle partition : Début » pour que la partition soit créée au début de l'espace libre.

Choisissez le système de fichiers **ext4**.

Enfin, choisissez le point de montage `/`. Je n'entrerai pas dans le détail du point de montage, il est trop tôt pour vous expliquer cela, mais sachez qu'en gros il permet d'indiquer le dossier dans lequel la partition sera créée (`/` étant le dossier racine, un peu comme `C:\` sous Windows).

Validez. La fenêtre principale se met à jour avec les nouvelles informations.

Étape 3 : créer une partition pour les documents

Cliquez à nouveau sur l'espace libre et rajoutez une nouvelle partition de la même manière (figure suivante).



Création de la partition des documents

Cette fois, vous pouvez créer une partition bien plus grande. Ce sera la partition où vous stockerez vos documents, un peu comme le « Mes documents » de Windows qui est souvent vite rempli de musiques et de films gourmands en espace disque.

Choisissez la taille que vous voulez pour cette partition, mais veillez à laisser à peu près 1 Go (environ 1 000 Mo) de libre sur votre disque pour que l'on puisse créer une dernière partition après.

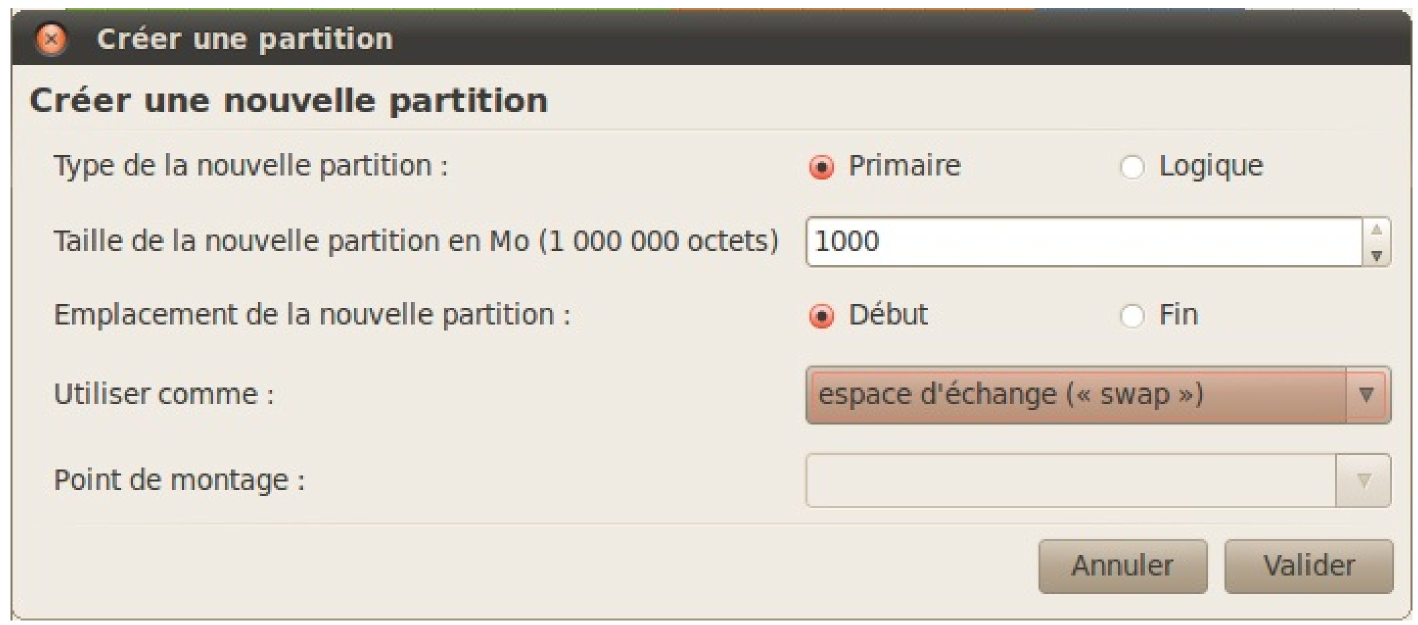
Choisissez là encore le système de fichiers **ext4**.

Pour le point de montage, choisissez `/home` (c'est le dossier « Mes documents » équivalent de Linux).

Étape 4 : créer une partition pour le swap

Il faut enfin créer une partition d'environ 1 Go appelée « swap ». C'est une partition un peu spéciale dont je ne vous ai pas parlé jusqu'ici pour ne pas vous embrouiller.

Pour faire simple, il s'agit d'une extension de la mémoire vive sur votre disque dur. Lorsque votre mémoire vive est pleine, Linux continue à fonctionner mais passe par le disque dur, grâce à la partition « swap ».

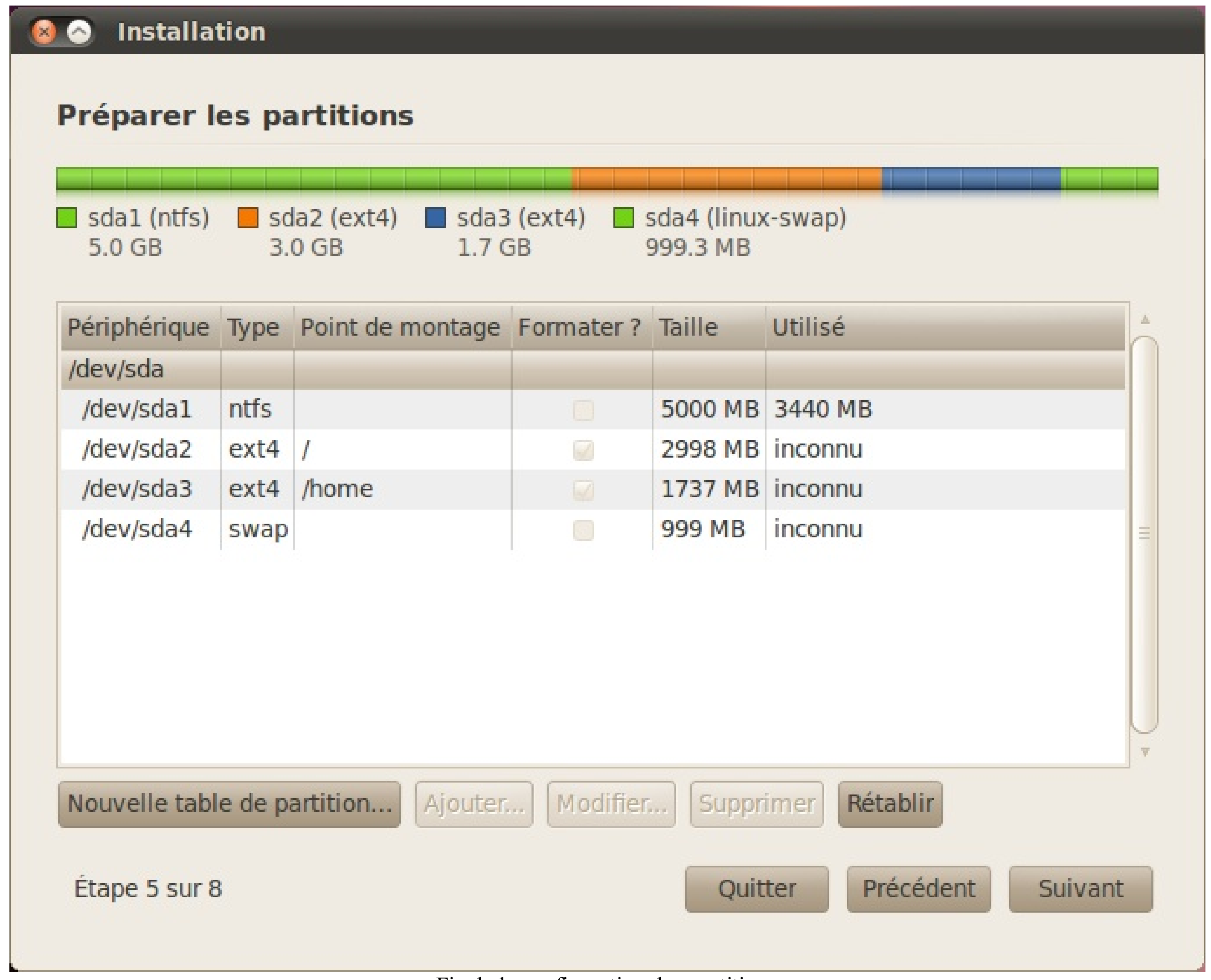


Création de la partition swap

Laissez tout l'espace libre restant pour cette partition, mais sachez que ça ne sert à rien qu'elle fasse plus d'1 Go en général.

Surtout, pensez à sélectionner le type « swap » (figure suivante). Vous n'aurez pas besoin de préciser de point de montage, contrairement aux autres fois.

Le résultat final est visible sur la figure suivante.



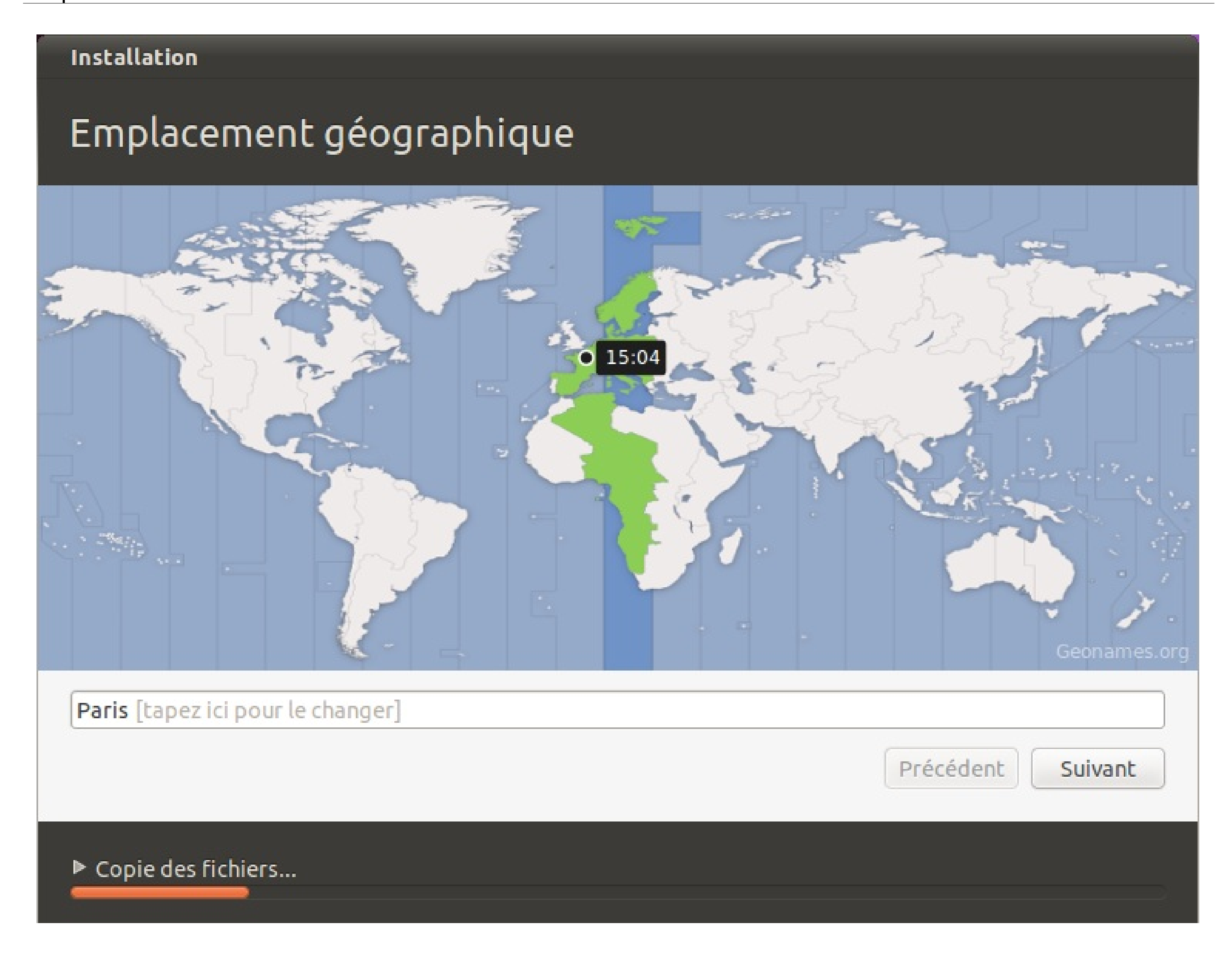
Fin de la configuration des partitions

Votre disque dur est configuré. Cliquez sur « Suivant ». :)

La fin de l'installation

Il nous reste encore quelques petites étapes et nous aurons terminé.

Sélection du fuseau horaire

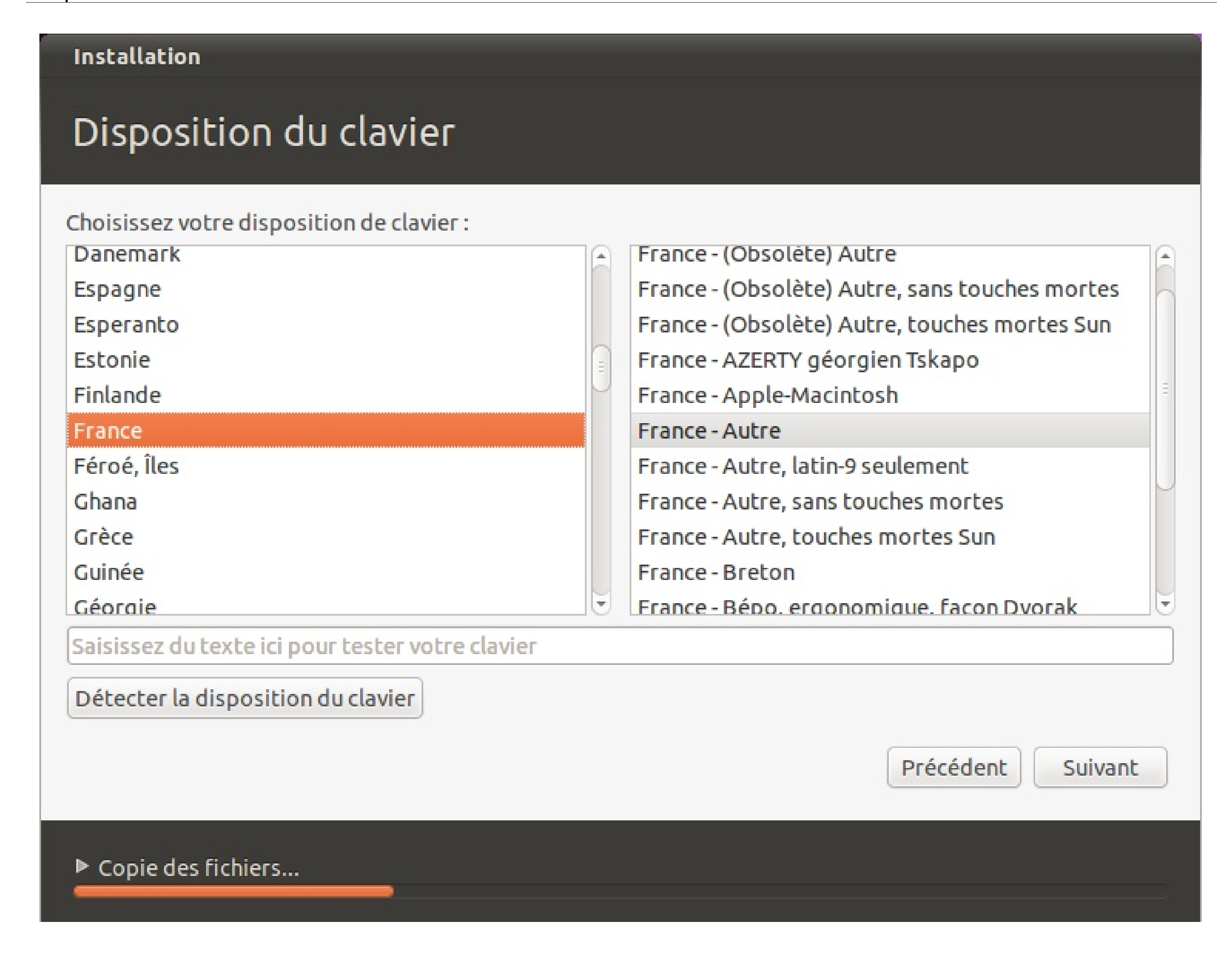


Comme vous pouvez le constater, l'installation s'effectue en tâche de fond (en bas de la fenêtre) pendant qu'on vous demande de régler quelques paramètres. Avec Ubuntu, on ne perd pas de temps !

On vous demande sur cet écran près de quelle grande ville vous habitez pour régler le fuseau horaire. Cliquez sur la carte sur le point correspondant à la ville la plus proche.

Vérifiez bien que l'heure indiquée est la bonne.

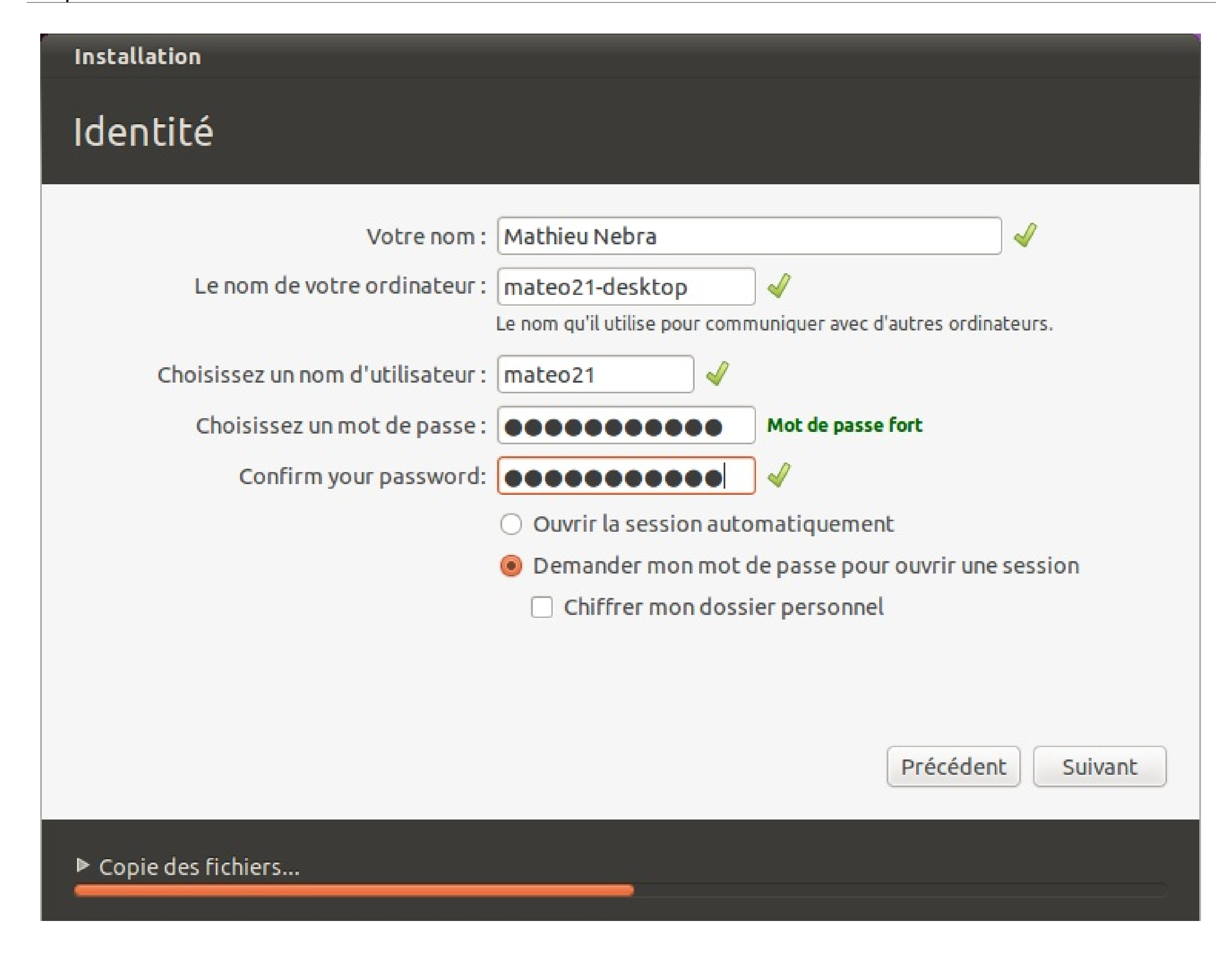
Le type de clavier



Dans la fenêtre qui suit, on vous demande quel type de clavier vous utilisez. Si vous habitez en France, vous avez un clavier dit « AZERTY », mais il se peut que vous habitiez un pays qui possède un clavier différent, comme la Suisse ou le Canada.

Pour les français, normalement le bon type de clavier est sélectionné d'office. Vous pouvez tester dans le petit cadre de texte en bas votre clavier. Essayez de taper des symboles "spéciaux" comme `é à ô ï` etc. Si ces symboles s'affichent sans problème, c'est que vous avez indiqué le bon type de clavier. :)

Choix du nom d'utilisateur



La fenêtre suivante (figure suivante) vous demande votre nom ainsi qu'un login (pseudonyme) qui vous identifiera sur votre ordinateur. Choisissez aussi un mot de passe.

En bas, on vous demande le nom que vous voulez attribuer à votre ordinateur. On vous en propose un par défaut mais vous pouvez changer cela sans risque.
Dans mon cas, comme je suis affreusement en manque d'inspiration, je vais laisser « mateo21-desktop » comme nom d'ordinateur.

Importation des données de Windows

Il se peut que le gestionnaire d'installation vous propose de récupérer quelques informations depuis Windows (figure suivante).

Installation

Transférer des documents et des paramètres

Veuillez sélectionner le ou les comptes à importer. Les documents et les paramètres de ces comptes seront disponibles dès la fin de l'installation.

Si vous ne souhaitez pas importer de comptes utilisateurs, ne sélectionnez rien et passez à la page suivante.

- Administrator *Microsoft Windows XP Professional (sda1)*
  - Internet Explorer
  - Wallpaper
  - My Documents
  - My Music
  - My Pictures

Étape 7 sur 8 — Quitter — Précédent — Suivant

Sélectionnez ce que vous souhaitez récupérer (par exemple votre fond d'écran), puis continuez.

Installation

Ouf ! C'est fini !
Votre travail à vous est terminé, vous avez indiqué toutes les informations nécessaires. L'installation se poursuit ensuite si elle n'était pas déjà terminée. Des écrans de présentation vous permettent d'en apprendre plus sur Ubuntu pour que vous ne vous ennuyiez pas !
On vous proposera de redémarrer pour finaliser l'installation de Linux. C'est bon, bravo, Linux est installé ! 



Juste avant l'extinction du PC pour le redémarrage, un message vous demandera de retirer le CD d'installation du lecteur et d'appuyer ensuite sur la touche Entrée de votre clavier. Cette précaution permet de s'assurer qu'Ubuntu démarrera bien à partir du disque dur (et non du CD !).

GRUB au démarrage

Lors de chaque démarrage, si vous avez choisi d'installer Linux sur le même disque dur que Windows, on vous demandera quel OS vous voulez charger (figure suivante).

```
GNU GRUB  version 0.95  (638K lower / 186304K upper memory)

 Ubuntu, kernel 2.6.8.1-3-386
 Ubuntu, kernel 2.6.8.1-3-386 (recovery mode)
 Memory test
 Other operating systems:
 Windows NT/2000/XP

    Use the ↑ and ↓ keys to select which entry is highlighted.
    Press enter to boot the selected OS, 'e' to edit the
    commands before booting, or 'c' for a command-line.
```

Ça, c'est GRUB, le programme qui permet de choisir l'OS à lancer au démarrage (souvenez-vous, je vous en ai parlé dans le premier chapitre !).
Il y a plusieurs options, mais ne paniquez pas, c'est très simple.

- **Ubuntu** : choisissez la première ligne pour démarrer Ubuntu, c'est-à-dire Linux.
- **Ubuntu (recovery mode)** : si vous avez des problèmes au lancement d'Ubuntu, utilisez ce « mode de récupération » pour résoudre ces problèmes. N'utilisez ce mode qu'en cas de nécessité absolue.
- **Memory Test** : pour faire un test de mémoire au cas où vous suspectez que votre mémoire vive est défaillante. Attention : le test est long, là encore ne le faites que si vous en avez vraiment besoin.
- **Other Operating Systems : Windows** : pour démarrer Windows.

Vous utiliserez les flèches de votre clavier pour sélectionner l'OS qui vous intéresse, et vous taperez `Entrée` pour valider.
Si vous mettez trop de temps avant de vous décider, GRUB lancera l'OS sélectionné.



Si vous avez installé Ubuntu sur un ordinateur équipé de Windows Vista ou Windows 7 et que vous constatez que celui-ci ne démarre plus, sachez qu'il s'agit d'un problème facile à régler. La solution se trouve ici.

En résumé

- Un disque dur peut être virtuellement découpé en plusieurs sections différentes : c'est le principe du **partitionnement**.
- Pour installer Linux, vous devez modifier le partitionnement de votre disque. L'opération consiste à réduire l'espace alloué à Windows pour faire de la place à Linux.
- Les partitions Windows stockent les fichiers selon un système appelé NTFS, tandis que les partitions sous Linux utilisent plus souvent les systèmes ext3 et ext4.
- On crée une partition spéciale appelée *swap* d'environ 1 Go, qui sert d'extension à la mémoire vive lorsque celle-ci est pleine.
- Après l'installation, un outil appelé GRUB vous demandera à chaque démarrage quel système d'exploitation vous souhaitez lancer.

 Découverte du bureau Unity

Nous y voilà enfin ! Après avoir découvert ce qu'est Linux et appris comment l'installer, vous devriez à présent avoir cet OS opérationnel sur votre ordinateur. Parfait. Vous l'avez installé, et maintenant ?

On n'installe pas Linux juste pour la beauté du geste et pour dire « ça y est, je suis sous Linux ! ». Si vous ne savez pas vous en servir, Linux risque davantage de faire office de décoration qu'autre chose sur votre ordinateur.

Toute la suite de ce cours sera donc dédiée à la découverte et l'étude approfondie de Linux. Vous n'imaginez pas encore toutes les choses que vous allez apprendre. ;)

Nous allons commencer en douceur par la découverte de Unity. C'est le gestionnaire de bureau par défaut de Ubuntu. Simple, clair et facile à prendre en main, il conviendra à la plupart des nouveaux utilisateurs de Linux.

Ce chapitre vous propose une petite visite guidée de l'environnement Unity.

Bienvenue sur le bureau Unity

Ce chapitre suppose soit :

- que vous avez installé Ubuntu (avec Unity par défaut) dès le début ;
- que vous avez installé autre chose (Kubuntu, Xubuntu) puis le paquet **ubuntu-desktop** pour obtenir Unity à l'aide du gestionnaire de programmes.

Lors du démarrage d'Ubuntu, vous allez être accueillis par une fenêtre de login (figure suivante). Cette fenêtre vérifie votre identité en vous demandant votre identifiant puis votre mot de passe.

Connexion à Ubuntu

Bon, le principe est simple, vous ne devriez pas avoir trop de mal : vous devez rentrer votre login et votre mot de passe. Ce sont les informations que vous avez indiquées lors de l'installation d'Ubuntu.

 Mais pourquoi faut-il s'authentifier à chaque fois que l'on démarre Linux ? Si je suis seul, ce n'est pas la peine de mettre


un mot de passe…

Il est vrai que sous Windows on n'a pas trop l'habitude de s'authentifier, surtout sur son ordinateur domestique… quoique les habitudes sont en train de changer de ce côté-là.

Sous Linux, la conception est totalement différente. C'est un OS qui se veut vraiment multiutilisateur, c'est-à-dire que plusieurs personnes peuvent utiliser le même ordinateur de façon simultanée (en le contrôlant à distance par l'internet par exemple). Il y a une vraie politique de sécurité et c'est pour cela que même pour l'ordinateur de la maison chacun doit avoir son login et son mot de passe. Cela permet notamment de savoir à qui appartient tel ou tel fichier.

Les options

Quelques options vous sont proposées si vous cliquez sur l'icône Ubuntu à côté de votre login. C'est là notamment que vous pouvez sélectionner le gestionnaire de bureau à lancer.

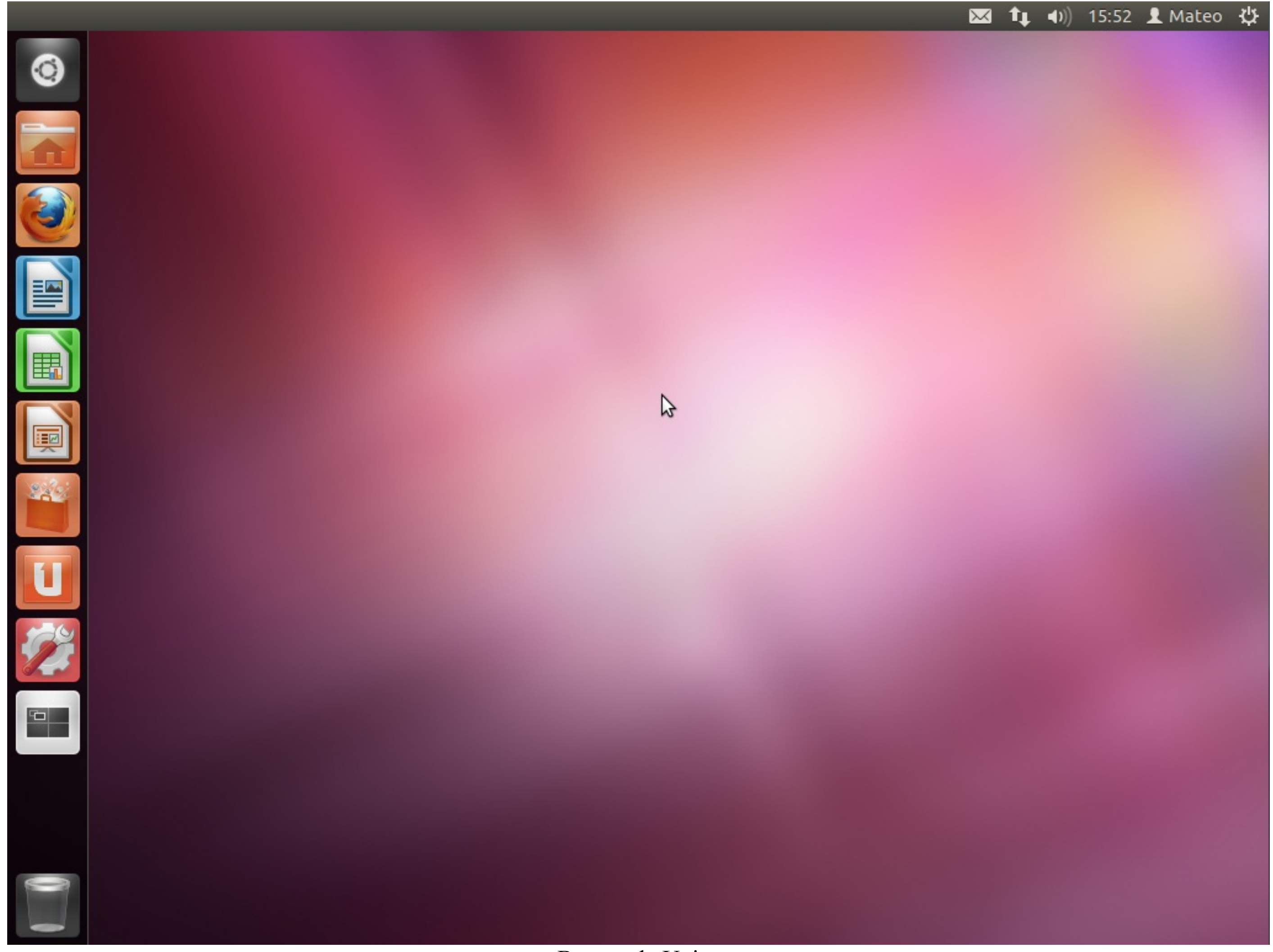


Choix du gestionnaire de bureau

Par défaut, seul Unity devrait être installé ("Ubuntu") et une version simplifiée ("Ubuntu 2D). Par la suite, si vous installez d'autres gestionnaires de bureau, vous pourrez sélectionner celui que vous souhaitez utiliser à ce moment-ci.

Présentation du bureau Unity

Une fois connectés, vous arrivez sur le bureau d'Unity, comme le montre la figure suivante.



Bureau de Unity

Le bureau est assez vide d'icônes, ce qui n'est pas plus mal pour ceux qui ont horreur des bureaux encombrés.

En fait, c'est un peu la philosophie d'Unity : une simplicité pour une meilleure esthétique. Vous allez vous rendre compte de tout cela par vous-mêmes.

La barre Unity à gauche

Commençons par la barre à gauche. Très importante, vous pouvez y lancer de nouvelles applications et afficher les fenêtres déjà ouvertes. C'est une barre qui ressemble beaucoup à la barre des tâches de Windows 7 ou au dock de Mac OS X.


Barre Unity

La première icône tout en haut ouvre le tableau de bord.

Le tableau de bord

De là, vous pouvez lancer tous vos programmes. Je vous laisse le parcourir un peu, l'ensemble est plutôt intuitif vous allez voir !


Vous pouvez lancer n'importe quel programme en tapant simplement son nom. Essayez de taper "Firefox" pour voir !

Les autres icônes de la barre Unity permettent d'ouvrir vos programmes favoris. Vous pouvez ajouter et retirer des icônes dans ce menu : les options apparaissent lorsque vous faites un clic droit.

Enfin, l'icône "Espaces de travail" permet de changer de bureau. En effet, vous avez 4 bureaux disponibles sous Ubuntu ! Si vous avez beaucoup de fenêtres ouvertes, cela peut vous permettre de mieux vous organiser.

La barre en haut

Elle donne accès aux menus de la fenêtre ouverte. Pointez avec la souris pour les faire apparaître (si vous avez une fenêtre ouverte).



La barre du haut

Sur le côté droit de cette barre, vous retrouvez quelques icônes de notification ainsi que la date. Enfin, un bouton en haut à droite vous propose d'arrêter votre ordinateur, ou bien de le redémarrer, de changer d'utilisateur, etc.

Nautilus, l'explorateur de fichiers

Nautilus est un explorateur de fichiers. C'est un programme du même type que l'explorateur de Windows.

Comme tout bon explorateur de fichiers qui se respecte, il vous permet de parcourir les dossiers et fichiers de votre disque dur et de les ouvrir. C'est donc un programme que vous risquez de lancer souvent.

Pour ouvrir Nautilus, cliquez sur l'icône dans la barre Unity :


Icône Nautilus

Nous nous trouvons ici dans le dossier personnel "Home", l'équivalent de "Mes documents" sous Linux.

Nautilus ouvre le dossier Home

Cette fenêtre est simple:

1. La première zone indique le chemin du dossier dans lequel vous vous trouvez, c'est-à-dire le nom du répertoire que vous êtes en train de visualiser. Chaque dossier est représenté par un bouton (figure suivante) et vous pouvez cliquer sur l'un des dossiers parents pour revenir en arrière.
2. Sur la gauche, une petite barre de raccourcis vous permet d'accéder à certains dossiers courants comme votre dossier personnel, le bureau, le lecteur CD, etc.
3. Enfin, la partie centrale affiche les fichiers et dossiers proprement dits.

Nautilus est donc un logiciel tout simple vous permettant de consulter les fichiers présents sur votre disque dur mais également sur des CD ou DVD. Son utilisation devrait vous être familière tant il ressemble à l'outil de Windows.

Vous mettrez un peu de temps à vous faire à l'organisation des dossiers qui est un peu particulière sous Linux, mais vous finirez par prendre vos repères. Pour le moment, je vous conseille d'utiliser votre répertoire personnel ; vous pouvez y stocker tous vos documents, vos vidéos, votre musique, etc.

Gestion des programmes

Ajout et suppression des programmes

L'ajout et la suppression de programme est simple et intuitive. Rendez-vous dans la logithèque Ubuntu en cliquant sur cette

icône :



Icône de la logitèque

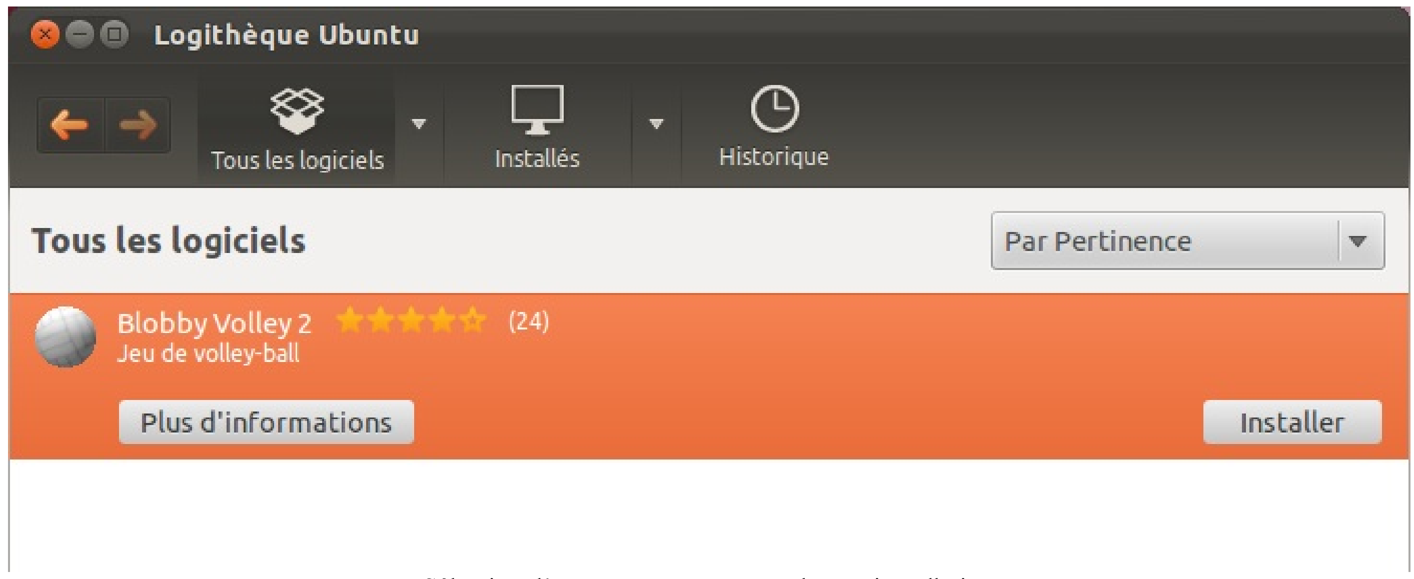
La fenêtre principale s'ouvre alors :

La logitèque Ubuntu

La logithèque Ubuntu est vraiment simple et agréable à utiliser. Elle fait assez penser à l'App Store des iPhone : les applications sont classées par catégories et vous pouvez les télécharger d'un simple clic.

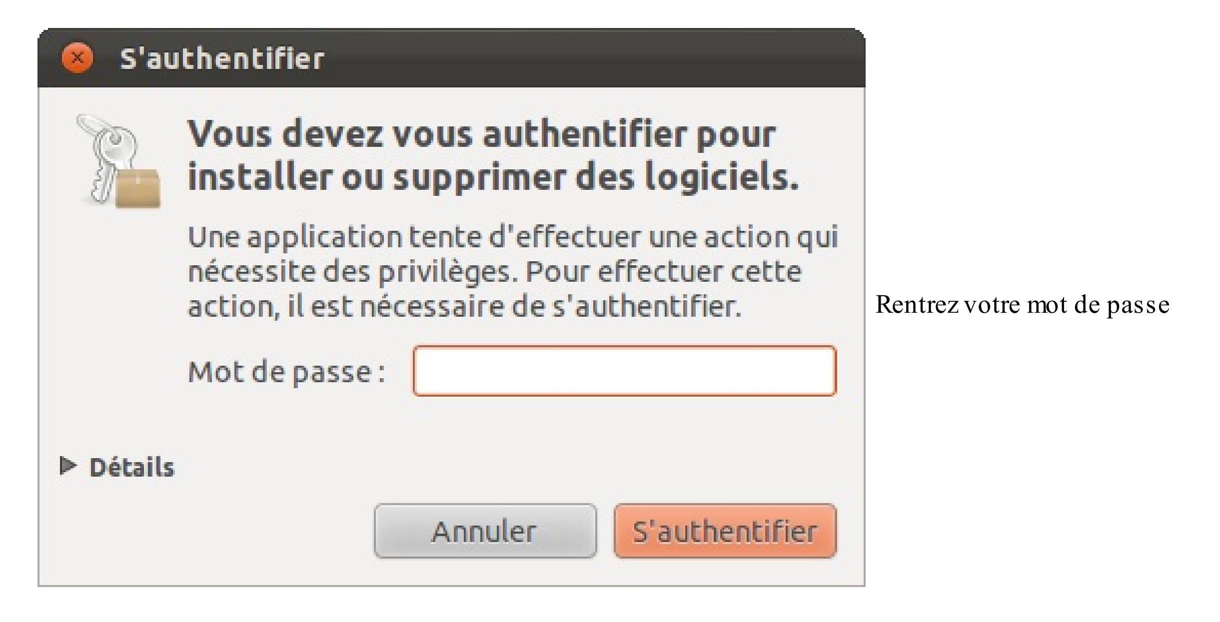
Commencez par faire un tour dans la section « Applications phares » qui vous propose une sélection des meilleures applications n'étant pas encore installées sur votre ordinateur. N'hésitez pas à en installer quelques-unes, vous y trouverez à coup sûr des programmes très intéressants.

Pour voir davantage d'applications, revenez en arrière et sélectionnez une catégorie (vous pouvez aussi faire une recherche via le champ en haut à droite). Je vais par exemple aller y chercher le jeu Blobby Volley 2.



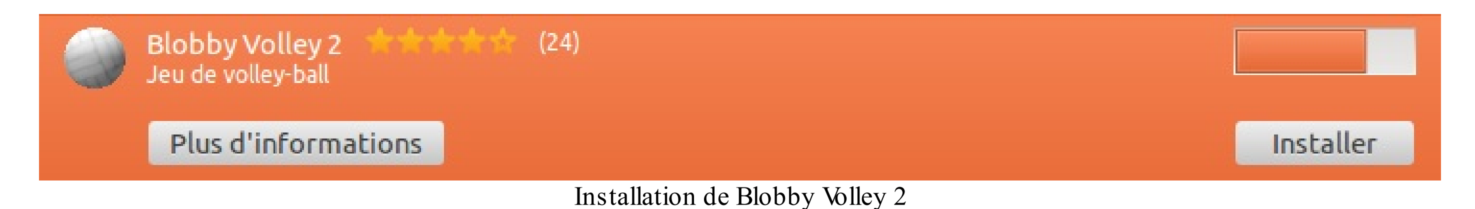
Sélection d'un programme en vue de son installation

Cliquez sur le bouton « Installer » (figure suivante). On vous demande à nouveau votre mot de passe par sécurité (figure suivante) ; il s'agit de celui que vous utilisez pour vous connecter au lancement de Linux.



Rentrez votre mot de passe

Il n'y a plus qu'à attendre que tout se fasse pour vous (figure suivante) !

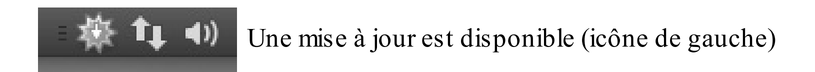


Installation de Blobby Volley 2

Mise à jour des programmes

Pouvoir ajouter et supprimer des programmes, c'est bien, mais il faut aussi les mettre régulièrement à jour pour profiter des nouvelles fonctionnalités et, surtout, corriger les failles de sécurité qui sont parfois détectées.

Vous êtes automatiquement notifiés dès que des mises à jour sont disponibles ; il suffit pour cela de regarder la petite icône en haut à droite de l'écran (à gauche sur la figure suivante).


Une mise à jour est disponible (icône de gauche)

Cliquez dessus pour afficher le détail des mises à jour, comme sur la figure suivante.

Fenêtre de mise à jour des programmes

Je vous conseille de ne pas vous prendre la tête et de tout laisser coché. Cliquez simplement sur **Installer les mises à jour** et laissez le logiciel faire le reste. :)


Et les mises à jour ne peuvent pas se faire automatiquement, sans que j'aie besoin à chaque fois de cliquer sur « Installer les mises à jour » ?

Si vous ne voulez pas vous prendre la tête et être sûrs d'avoir un système toujours à jour, le mieux est de configurer le gestionnaire de mises à jour pour qu'il installe les nouveautés sans demander votre autorisation.

Retournez dans le menu **Applications** → **Ajouter & Enlever**. Dans le bas de la fenêtre qui s'ouvre, cliquez sur **Préférences**. Cliquez ensuite sur l'onglet **Mises à jour** puis sélectionnez **Installer les mises à jour de sécurité sans confirmation** (figure suivante).

Configurez les mises à jour automatiques

Et voilà le travail !

En résumé

- Unity est un des plus célèbres gestionnaires de bureau. Il est installé par défaut avec Ubuntu.
- Vous devez vous connecter au démarrage de la machine, en indiquant votre nom d'utilisateur et votre mot de passe.
- Les menus en haut de l'écran sous Unity donnent accès à vos programmes et fichiers.
- La logithèque Ubuntu est un outil simple d'emploi qui vous permet d'installer de nouveaux programmes.
- Les mises à jour de tous les programmes sont centralisées et peuvent s'effectuer automatiquement.

 Découverte du bureau KDE

Après avoir découvert Unity, l'environnement de bureau par défaut d'Ubuntu, nous allons ici nous pencher sur KDE. Celui-ci est utilisé par défaut si vous installez Kubuntu (une variante d'Ubuntu) mais peut tout aussi bien être installé sous Ubuntu, comme nous allons le voir.

L'objectif est avant tout de vous montrer la diversité qui règne dans le monde de Linux : celui-ci peut prendre plusieurs formes selon le gestionnaire de bureau que l'on utilise. KDE est assez différent de Unity, vous allez vite vous en rendre compte, mais il est très plaisant à utiliser. Ce chapitre sera l'occasion de vous faire une première idée de KDE qui est, avec Unity, un des gestionnaires de bureau les plus utilisés et les plus célèbres.

Comment obtenir KDE ?

Pour essayer KDE, il faut avant toute chose l'installer. Tout dépend de votre cas.

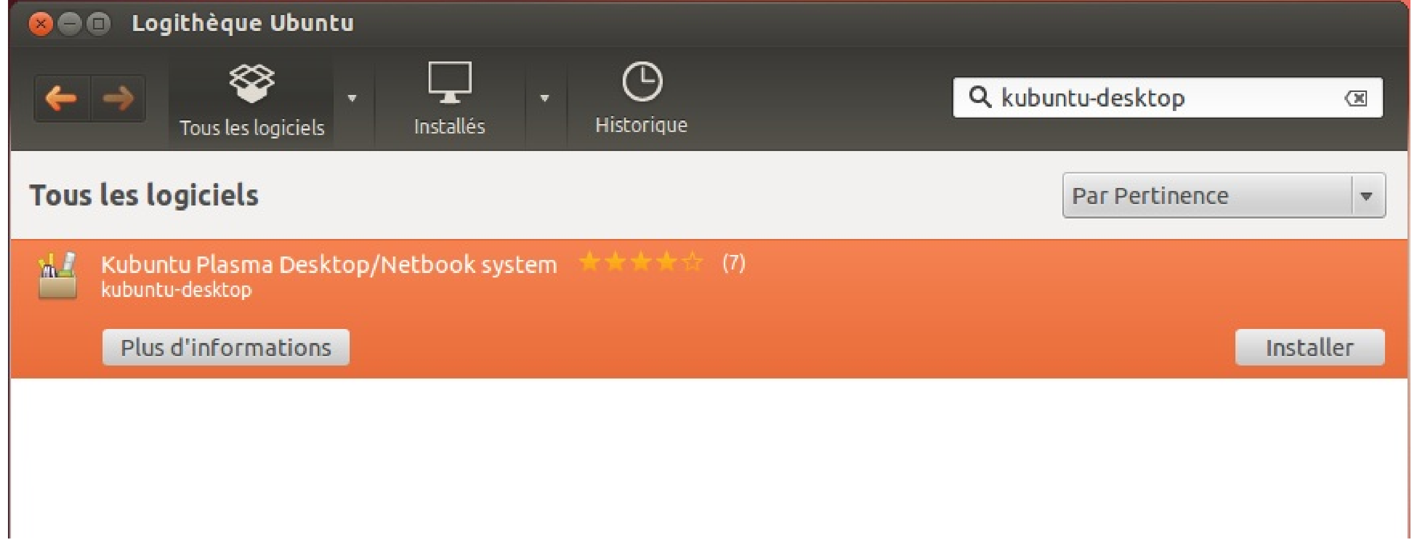
- Si vous avez téléchargé et installé Kubuntu (et non Ubuntu), KDE est installé par défaut. Vous n'avez donc rien à faire : vous serez automatiquement sous KDE !
- Si vous avez téléchargé et installé Ubuntu (ou Xubuntu), vous allez devoir installer KDE avant de continuer.

Si vous êtes sous Ubuntu et que vous voulez installer KDE, deux choix s'offrent à vous.

- **Installer le KDE « minimal »** : cela vous permettra d'obtenir KDE et les applications de base (navigateur, explorateur de fichiers). Pour cela, vous devez installer le programme `kde-minimal` (aussi appelé *L'environnement de bureau K, applications minimales*).
- **Installer KDE complet** : vous aurez KDE et toute une série d'applications dédiées à cet environnement. Pour cela, vous devez installer `kubuntu-desktop` (aussi appelé *Kubuntu Plasma Desktop System*).

Les applications de la version complète sont nombreuses et très intéressantes, bien qu'elles fassent parfois des doublons avec celles déjà installées. Dans les exemples qui vont suivre, je vais installer la version complète (équivalente à l'installation de Kubuntu). Cependant, dans un premier temps, vous pouvez tout aussi bien installer la version minimale si vous le désirez.

Ouvrez la logithèque Ubuntu comme vous avez appris à le faire. Dans le champ de recherche en haut à droite, tapez `kubuntu-desktop` puis installez le premier programme de la liste (figure suivante).



Installation de KDE

Patientez le temps de l'installation, cette dernière pouvant être un peu plus longue que d'habitude. Une fois cela fait il vous faudra soit redémarrer votre ordinateur, soit vous déconnecter de votre session.

Connexion au bureau KDE

La suite de ce chapitre suppose :

- soit que vous avez installé Kubuntu (avec KDE par défaut) dès le début ;
- soit que vous avez installé autre chose (Ubuntu, Xubuntu) mais que vous avez choisi le paquet `kubuntu-desktop` ou `kde-minimal` comme expliqué précédemment.

Pour accéder à KDE, tout dépend de votre cas.

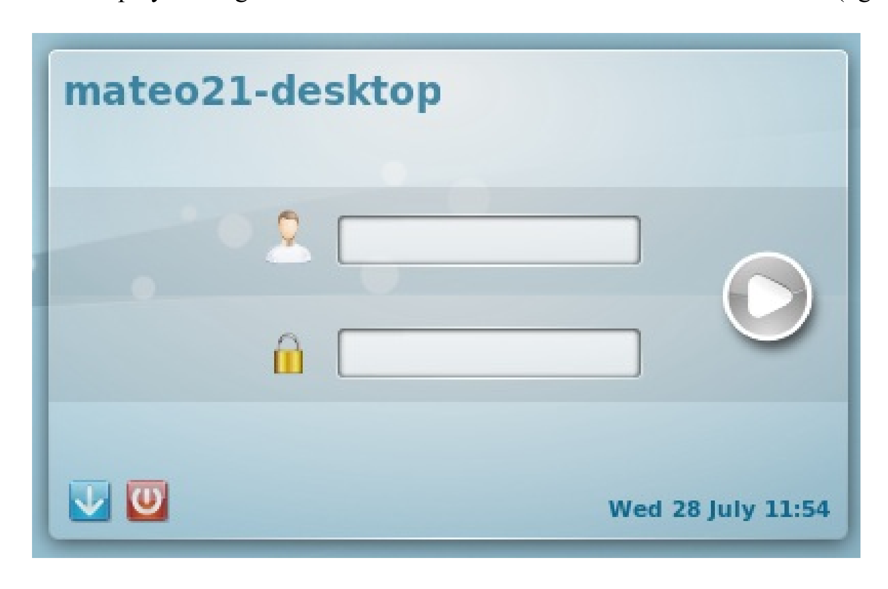
- Si vous avez installé Ubuntu puis que vous avez suivi la procédure indiquée précédemment pour installer KDE, vous devrez sélectionner ce dernier dans l'écran de login (voir le chapitre précédent).

- Si vous avez installé Kubuntu, la fenêtre de login est différente, mais ce sera bien KDE qui sera lancé.

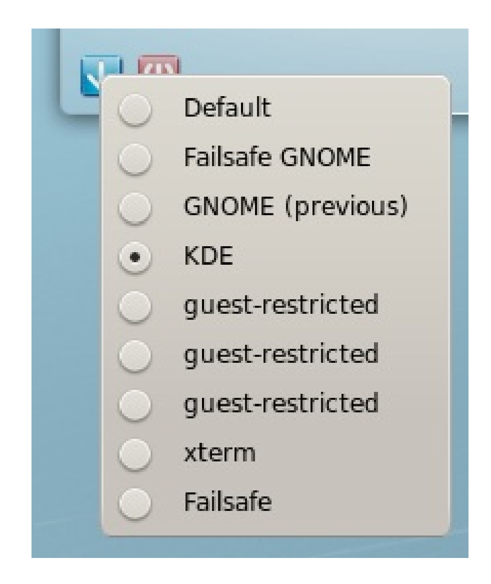
Je vais vous présenter l'interface de login de KDE, appelée KDM. Si vous avez toujours l'interface de login d'Unity, que nous avons découverte au chapitre précédent, sachez que cela ne change rien : vous pouvez très bien lancer KDE depuis l'interface de login d'Unity et vice-versa.

KDM, le programme de login de KDE (Kubuntu)

KDM est l'abréviation de KDE Display Manager. C'est l'interface de connexion aux couleurs de KDE (figure suivante).



Si vous cliquez sur la petite flèche pointant vers le bas vous pourrez sélectionner le gestionnaire de bureau que vous souhaitez lancer. Vérifiez que « KDE » est bien sélectionné, comme indiqué à la figure suivante.



Indiquez votre login et votre mot de passe. S'ils sont bons, KDE se charge alors (figure suivante).



Le bureau et le menu K

Après un petit instant de chargement, vous vous retrouvez sur le bureau de KDE, présenté sur la figure suivante.

Le bureau est assez riche au premier abord. En fait, il est complètement personnalisable : on peut y ajouter toute une variété de **widgets**, appelés « Plasmoïdes ». Vous pouvez les configurer en cliquant sur l'icône située tout en haut à droite de l'écran.

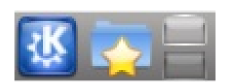
En bas de l'écran, on retrouve une barre des tâches qui rappelle en plusieurs points celle de Windows ; vous ne devriez pas être trop dépaysés. Sous KDE, on l'appelle le **tableau de bord**.

Le tableau de bord

Le tableau de bord, normalement présent en bas de l'écran, est un outil complet qui vous permet de lancer vos applications, d'accéder à vos fichiers ou encore de visualiser l'état du système. Intéressons-nous dans un premier temps à la partie gauche de ce tableau de bord.

Les boutons à gauche du tableau de bord

À gauche, on retrouve plusieurs icônes, présentées sur la figure suivante.



Analysons le rôle de ces boutons, de gauche à droite.

Le menu K

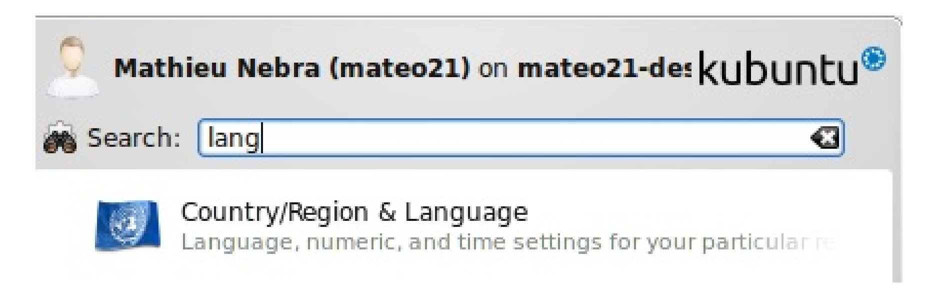
C'est le menu principal, le plus important de KDE (figure suivante).



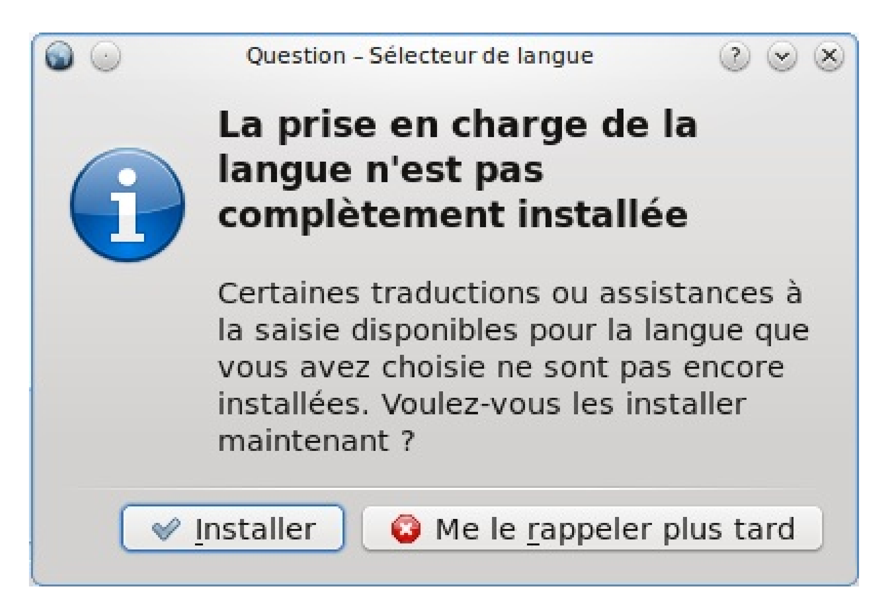
Il permet de lancer des applications, de configurer le système et d'arrêter l'ordinateur. Il ressemble d'ailleurs au menu `Démarrer` de Windows 7.

Le `menu` K s'ouvre sur vos applications favorites. Un clic droit sur l'une d'elles permet de l'ajouter ou de la retirer de vos favoris pour un accès plus rapide.

Si KDE est en anglais, recherchez le programme « Country & Region Language ». Vous pouvez tout simplement taper « lang » tandis que le menu K est ouvert (figure suivante).



Dans la fenêtre qui s'ouvre, cliquez sur « Select System Language » et installez les traductions lorsqu'on vous le demande (figure suivante).



Après un temps d'installation, on vous demandera de sélectionner votre langue : indiquez le français.
Il faudra ensuite vous déconnecter et vous reconnecter à KDE pour que les changements soient pris en compte.

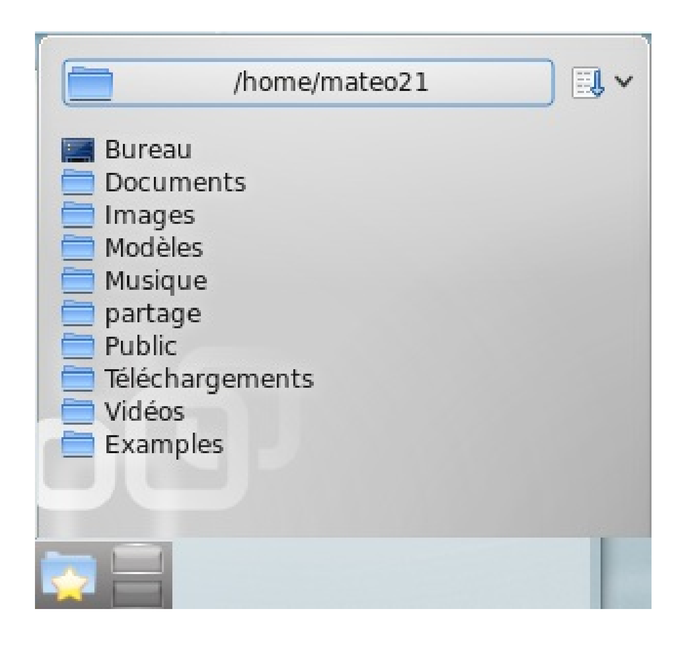
Continuez ensuite à explorer les programmes que vous pouvez lancer avec le menu K. Il vous est également possible d'accéder à votre poste de travail, vos documents récents, de vous déconnecter, d'arrêter l'ordinateur… Je vous laisse découvrir tout cela. ;-)

Jetez un coup d'œil en particulier à :

- **Dolphin** : le programme qui permet de parcourir vos fichiers ;
- **Konqueror** : le navigateur web (qui permet aussi de parcourir vos fichiers).

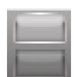
L'explorateur QuickAccess

Cette seconde icône du tableau de bord ouvre directement votre répertoire personnel (« home »). Vous pouvez l'utiliser pour accéder rapidement à vos fichiers (figure suivante) sans passer par le programme Dolphin.



Le gestionnaire de bureaux virtuels

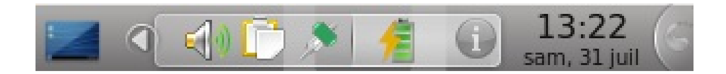
Comme sous Unity, vous pouvez avoir plusieurs bureaux virtuels différents (figure suivante). Cela vous permet de mieux vous organiser si vous avez beaucoup de fenêtres ouvertes.



Ici par défaut il n'y a que deux bureaux virtuels, mais vous pouvez en ajouter d'autres (clic droit → Configurer les bureaux virtuels…).

Les boutons à droite du tableau de bord

Passons maintenant à la droite du tableau de bord (figure suivante).



La première icône vous permet d'afficher à nouveau le bureau lorsque vous avez de multiples fenêtres ouvertes. Cela vous permet de réduire toutes les fenêtres d'un coup. Si vous cliquez une seconde fois, elles réapparaîtront à nouveau.

Ensuite, une série d'icônes sont présentes dans ce que l'on appelle la **zone de notification**. Le principe est exactement le même que sous Windows : certains programmes qui tournent apparaissent ici et vous informent des événements en cours, du volume audio, de l'état de la batterie, etc.

Enfin, vous avez la date et, tout à droite, une petite icône qui vous permet de personnaliser complètement le tableau de bord.

Voilà pour ce rapide tour d'horizon du tableau de bord. Bien sûr, ces icônes peuvent changer d'une version à l'autre d'Ubuntu et vous pouvez reconfigurer le tableau de bord comme bon vous semble. N'hésitez pas à le personnaliser jusqu'à ce que vous vous sentiez un peu plus « chez vous ». :-)

L'explorateur de fichiers Dolphin

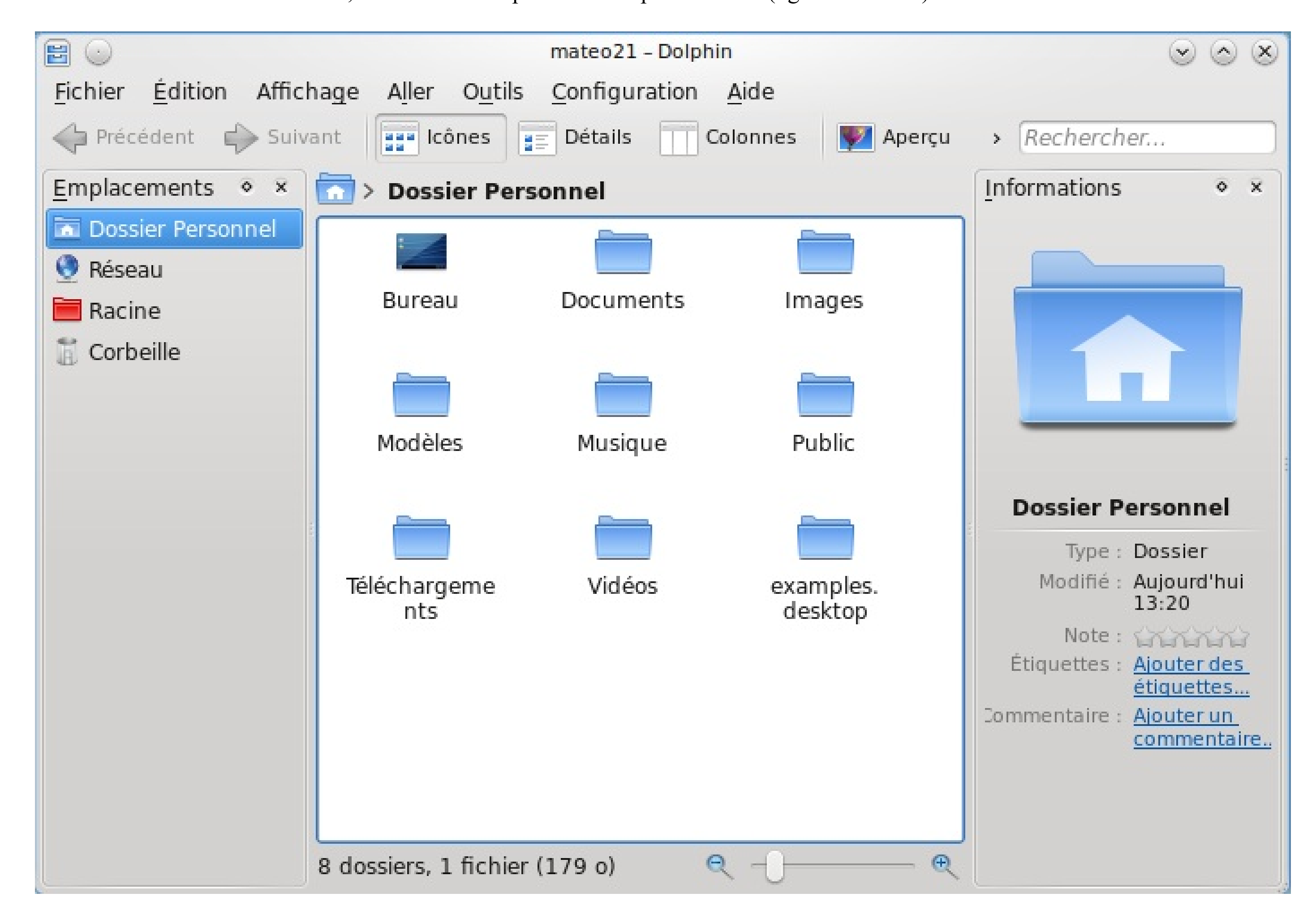
Pour ouvrir l'explorateur de fichiers, le programme qui vous permet de parcourir votre disque dur, plusieurs possibilités s'offrent à vous :

- ouvrir le `menu K` et cliquer sur Dolphin dans la liste ;
- ouvrir le `QuickAccess` (à côté du menu K), sélectionner un dossier et cliquer sur le bouton « Ouvrir » en haut.



Notez que le navigateur web de KDE, Konqueror, peut aussi jouer le rôle d'explorateur de fichiers.

Dans un cas comme dans l'autre, la fenêtre de l'explorateur Dolphin s'ouvre (figure suivante).



Son fonctionnement n'est, là encore, pas très différent de l'explorateur Windows. Après quelques minutes de découverte, vous vous serez vite adaptés !

La principale particularité de Dolphin (et de KDE en général) est qu'il faut cliquer une seule fois pour ouvrir un fichier ou un dossier. Inutile donc de double-cliquer à tout va !

Vous pouvez ouvrir plusieurs onglets (comme dans un navigateur web) en effectuant la combinaison de touches `Ctrl + T`. Mieux encore, vous pouvez scinder la fenêtre en deux (figure suivante) pour voir simultanément deux dossiers différents ! Pour cela, vous pouvez appuyer sur la touche `F3` ou cliquer sur le bouton « Scinder » en haut dans la barre d'outils. Ainsi, vous pouvez facilement déplacer ou copier des fichiers d'un dossier à l'autre !

Ces particularités mises à part, n'ayez crainte, vous trouverez vite vos marques sous KDE.



Je vous invite maintenant à ouvrir la « Configuration du système » (équivalent du « Panneau de configuration » de Windows) qui vous permettra de personnaliser au mieux votre KDE. Pour l'ouvrir, vous savez ce qu'il vous reste à faire : utiliser le menu K !



Si vous souhaitez installer des programmes sous KDE, lancez le programme « Ubuntu Software Center » ; il s'agit de la logithèque Ubuntu que nous avons découverte sous Unity.

En résumé

- KDE est un autre gestionnaire de bureau très célèbre, installé par défaut si vous avez téléchargé Kubuntu.
- Si vous avez choisi Ubuntu et que vous utilisez donc Unity, vous pouvez installer KDE en passant par la logithèque. Il suffit d'installer `kubuntu-desktop` (version complète avec de nouveaux programmes) ou `kde-minimal` (version allégée).
- Vous pouvez sélectionner votre gestionnaire de bureau au démarrage, sur l'écran de connexion.
- Le tableau de bord de KDE est entièrement personnalisable. Vous pouvez lancer vos programmes depuis le menu K, similaire au menu Démarrer de Windows.

 Installez Linux dans une machine virtuelle

La virtualisation est une technique de plus en plus répandue en informatique : cela consiste à faire tourner un ordinateur « virtuel » dans votre ordinateur. Imaginez : cela vous permet de lancer Linux à l'intérieur d'une fenêtre Windows ou Windows à l'intérieur d'une fenêtre Linux, voire pourquoi pas Linux dans une fenêtre Linux !

La virtualisation peut sembler assez impressionnante et complexe, mais elle est devenue accessible au grand public ces dernières années. Dans ce chapitre, nous allons apprendre à utiliser le logiciel libre Virtual Box pour installer Linux à l'intérieur de Windows.

Le gros avantage de cette technique est que Linux sera « isolé » dans la machine virtuelle : il ne risque absolument pas d'altérer le bon fonctionnement de Windows. Une technique à réserver à ceux qui souhaitent tester Linux sans aucun risque !

Dans ce chapitre, nous allons découvrir comment lancer Linux à l'intérieur d'une fenêtre Windows (figure suivante). C'est le principe de la virtualisation !



À la fin de ce chapitre, vous aurez appris à installer Linux dans une machine virtuelle au sein de Windows. C'est une opération qui peut sembler compliquée mais qui est en fait assez simple. Son gros avantage : vous ne risquez pas d'altérer le fonctionnement de votre ordinateur en installant Linux de cette façon.

Installer VirtualBox

Pour commencer, nous devons télécharger et installer un logiciel de virtualisation. Celui que je vous recommande s'appelle VirtualBox. Il a l'avantage d'être libre et gratuit, et il existe en version Windows, Linux et Mac OS X.

Rendez-vous sur le site de VirtualBox pour télécharger la dernière version correspondant à votre système d'exploitation.

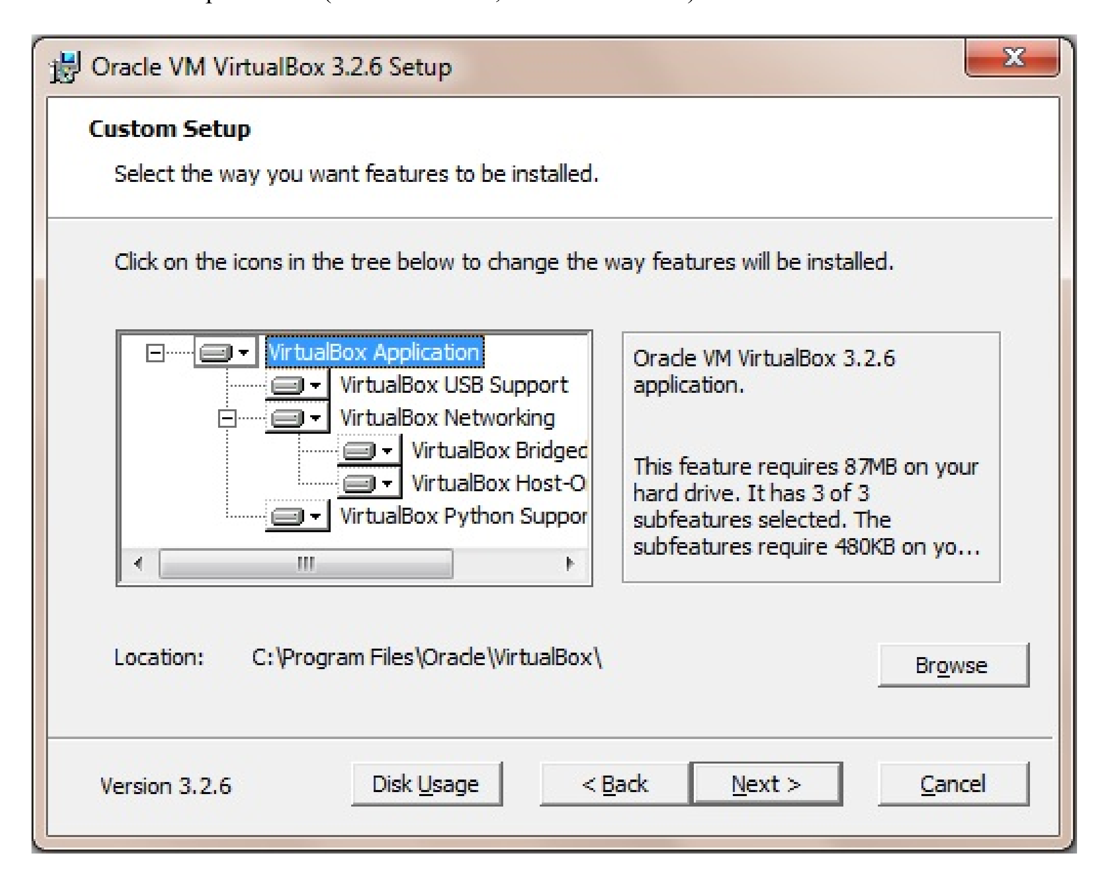


Si votre ordinateur est équipé de Windows, téléchargez VirtualBox pour Windows *même* si vous comptez l'utiliser pour installer Linux. Je reconnais que cette histoire « d'ordinateur dans l'ordinateur » peut prêter à confusion, donc imaginez tout simplement que vous allez installer un nouveau programme pour Windows. Si vous avez Mac OS X, téléchargez la version Mac OS X.

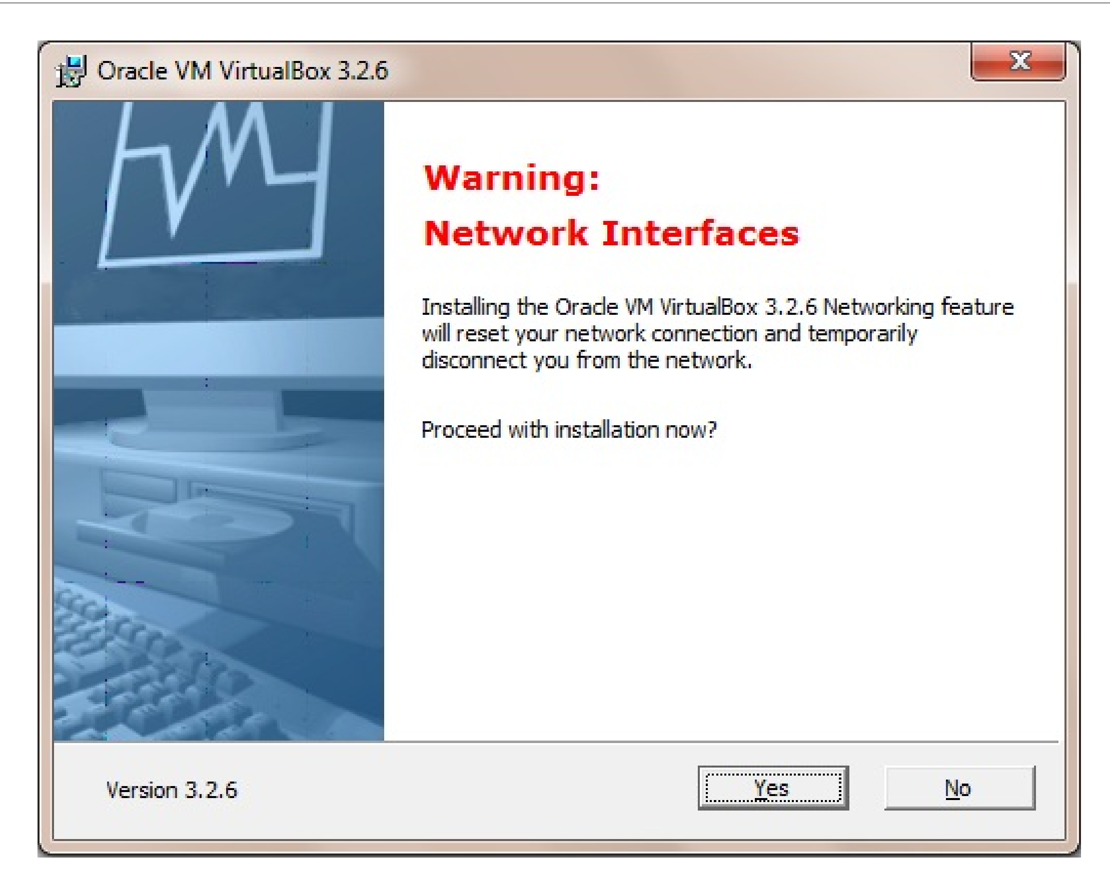
Lancez le programme d'installation (figure suivante).



L'assistant d'installation vous demande quels sont les éléments que vous souhaitez installer (figure suivante). Je vous recommande de laisser les choix par défaut (en l'occurrence, tout sera installé).



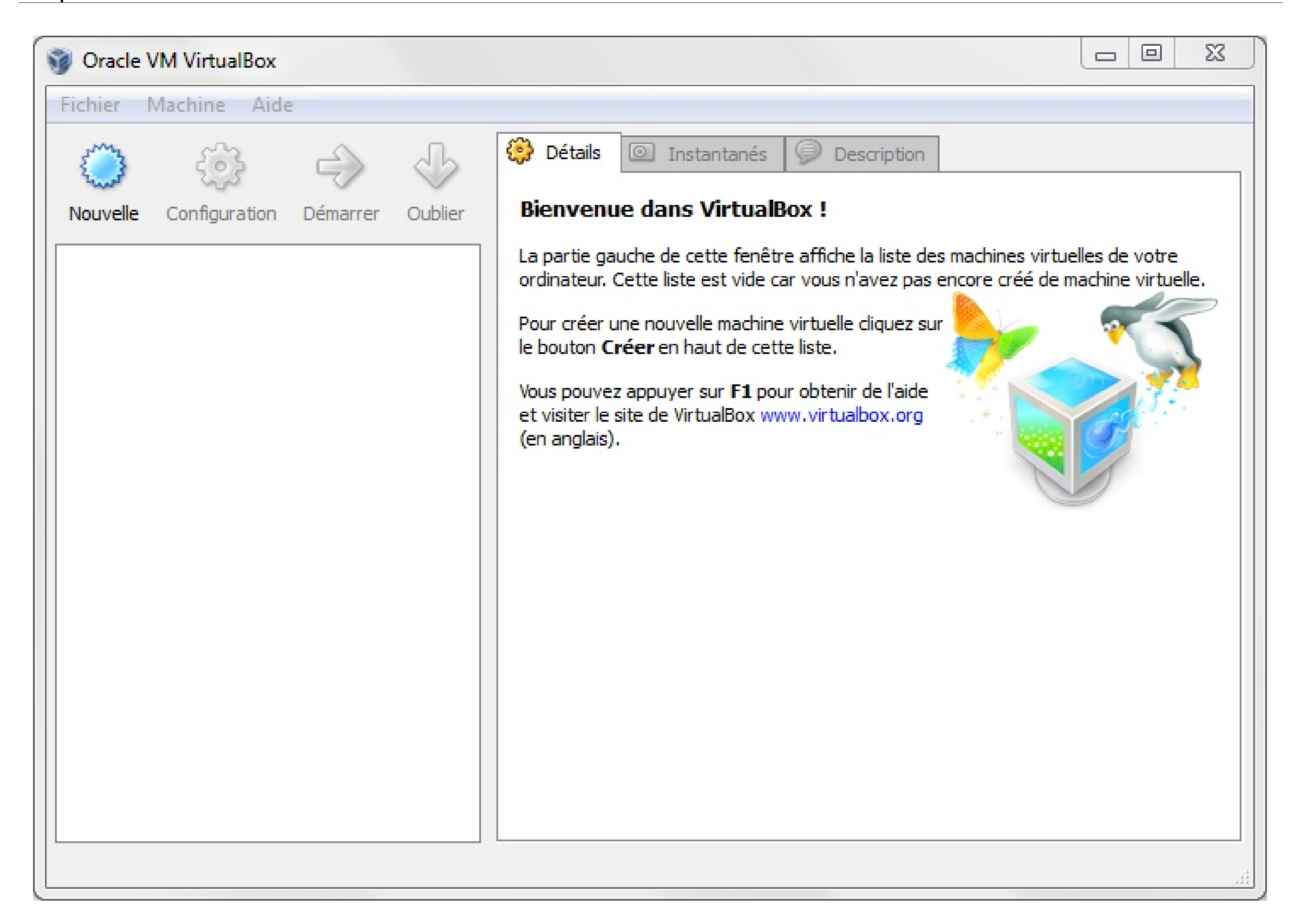
Avant de lancer l'installation, l'assistant vous prévient qu'il va devoir temporairement couper votre accès à Internet (figure suivante). En effet, VirtualBox doit établir un pont de connexion entre votre ordinateur et la machine virtuelle pour que celle-ci puisse accéder à Internet. Cela provoque une rupture temporaire de l'accès à Internet de l'ordre de quelques secondes (dans la plupart des cas ce n'est pas gênant, mais il est toujours plus agréable d'être prévenu :)).



L'installation débute ensuite ; vous n'avez rien à faire. À la fin, un nouveau programme nommé VirtualBox est installé. Il ne vous reste plus qu'à le lancer !

Créer une nouvelle machine virtuelle

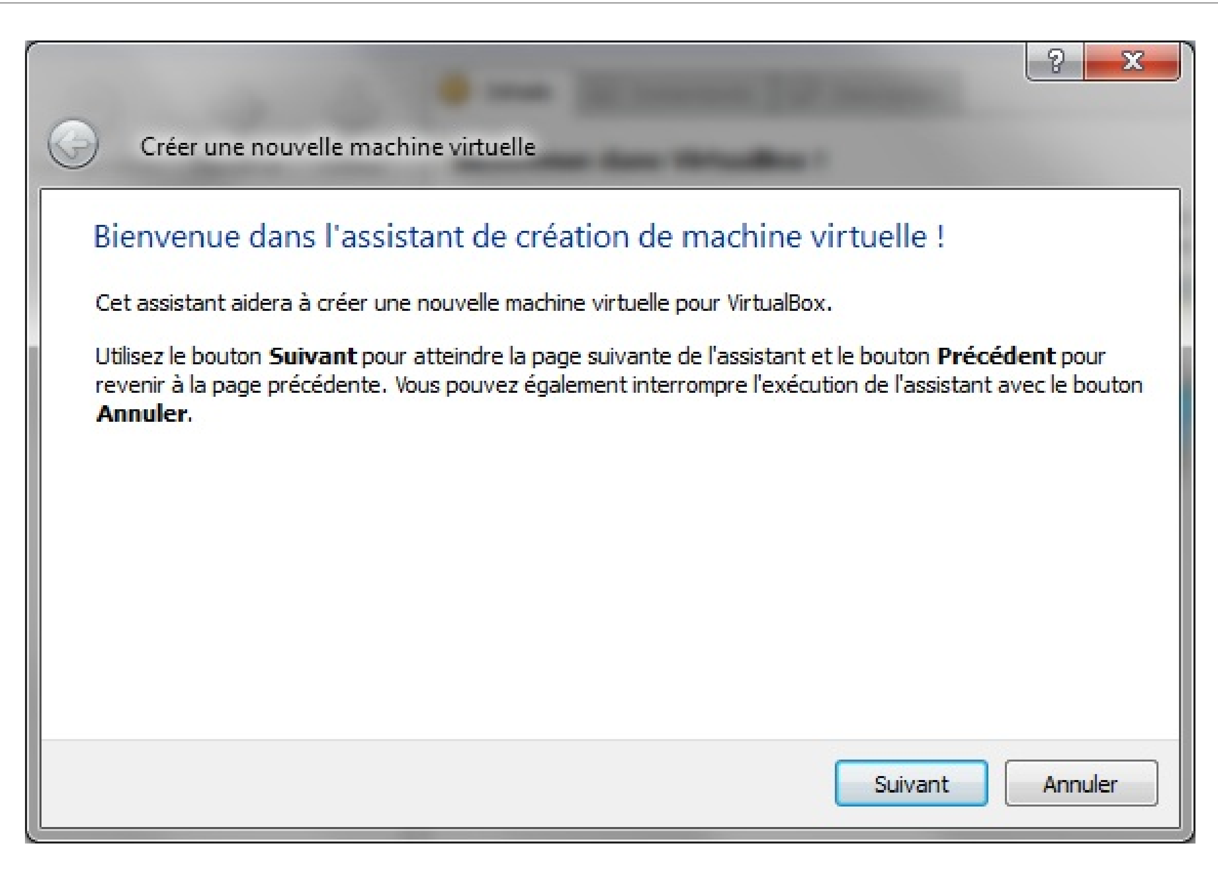
Lors de son lancement, VirtualBox affiche un écran semblable à la figure suivante.



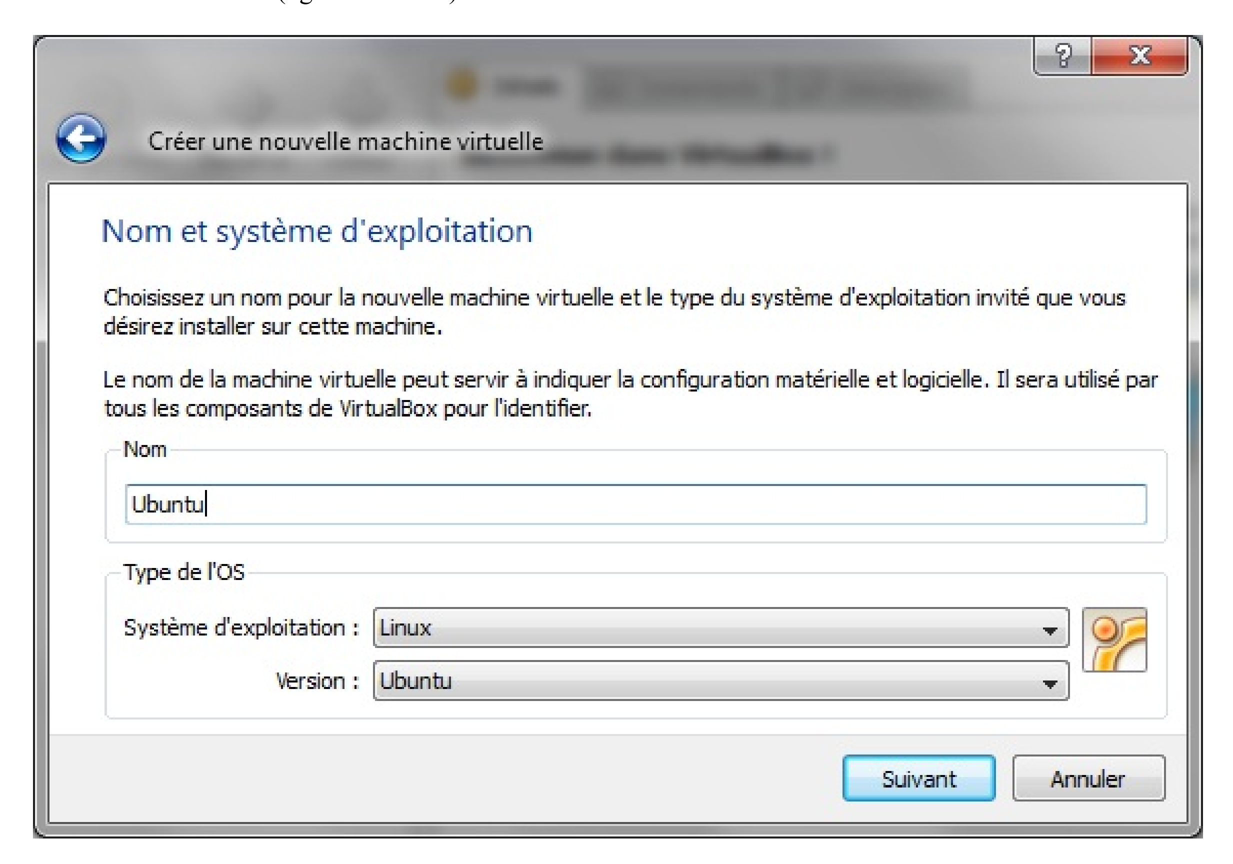
Dans ce programme, vous allez créer des machines virtuelles. Ce seront des miniordinateurs qui s'exécuteront à l'intérieur de Windows. Ils consommeront donc un peu de mémoire vive et d'espace disque et utiliseront votre processeur. C'est pourquoi il est recommandé d'avoir un ordinateur *assez* puissant pour utiliser VirtualBox (la plupart des ordinateurs d'aujourd'hui n'auront aucun problème pour faire tourner une machine virtuelle).

L'assistant de création de machine virtuelle

Nous devons commencer par créer une nouvelle machine virtuelle. Cliquez sur le bouton « Nouvelle » en haut à gauche. L'assistant de création de machine virtuelle apparaît (figure suivante).



On vous demande dans un premier temps de lui donner un nom. Vous pouvez tout simplement l'appeler « Ubuntu ». Vous remarquerez que les champs en dessous se remplissent automatiquement pour indiquer le type de système d'exploitation qui sera installé dans la machine virtuelle (figure suivante).



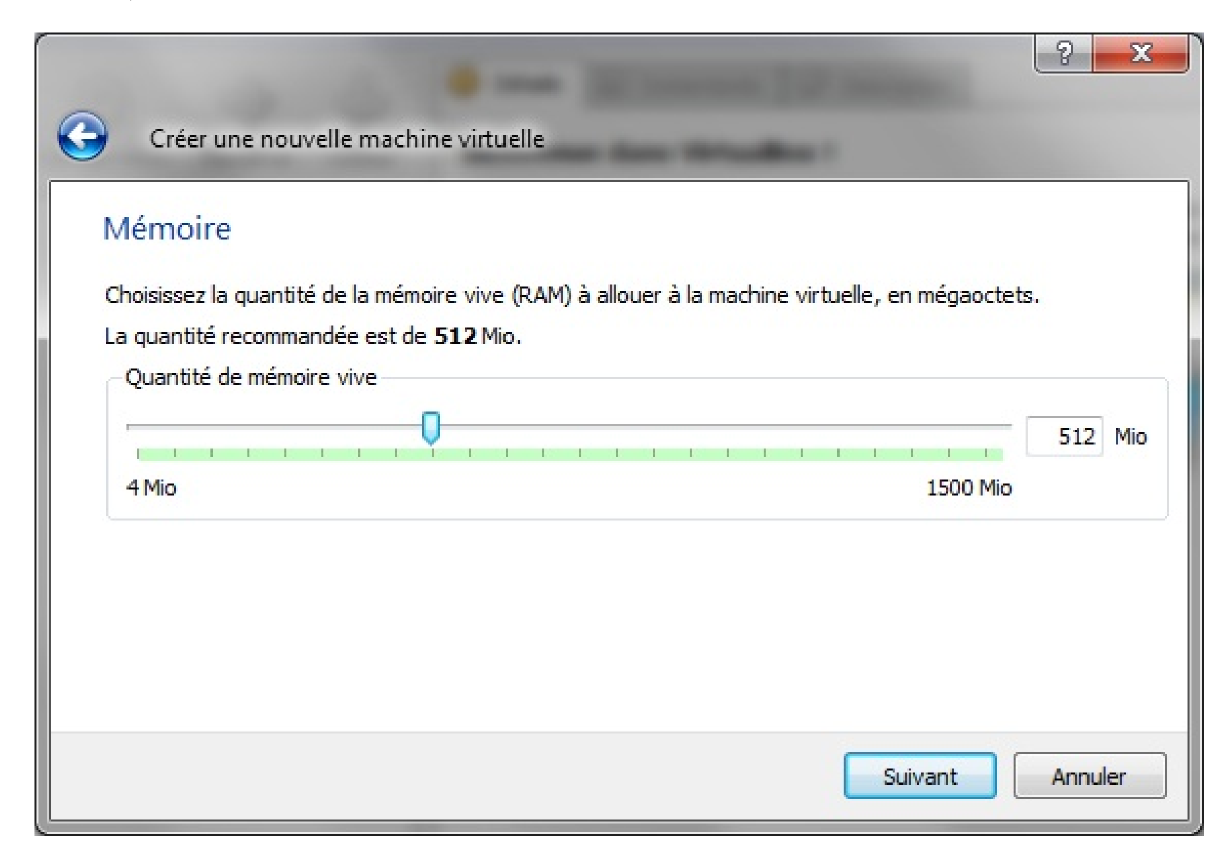
Vous constaterez que VirtualBox est capable de faire tourner de nombreux systèmes d'exploitation, de Windows 3.1 à Windows 7 en passant par Linux (Ubuntu, Red Hat, Debian…), Solaris, FreeBSD, etc.



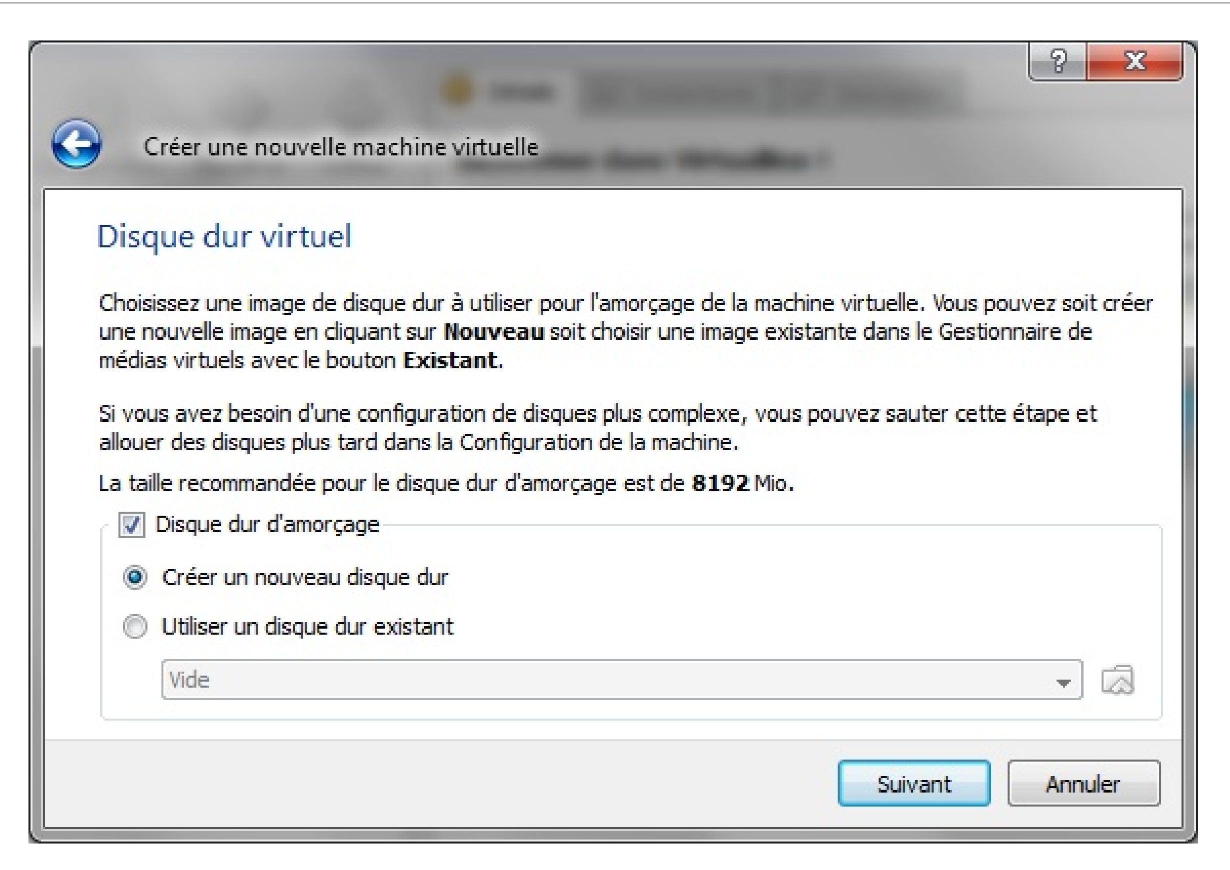
VirtualBox est *capable* de faire tourner tous ces systèmes d'exploitation, mais il vous faut le CD d'installation ou l'image disque de ces OS pour les lancer. VirtualBox n'est pas un outil magique : sans le CD d'installation, il ne saura pas quoi

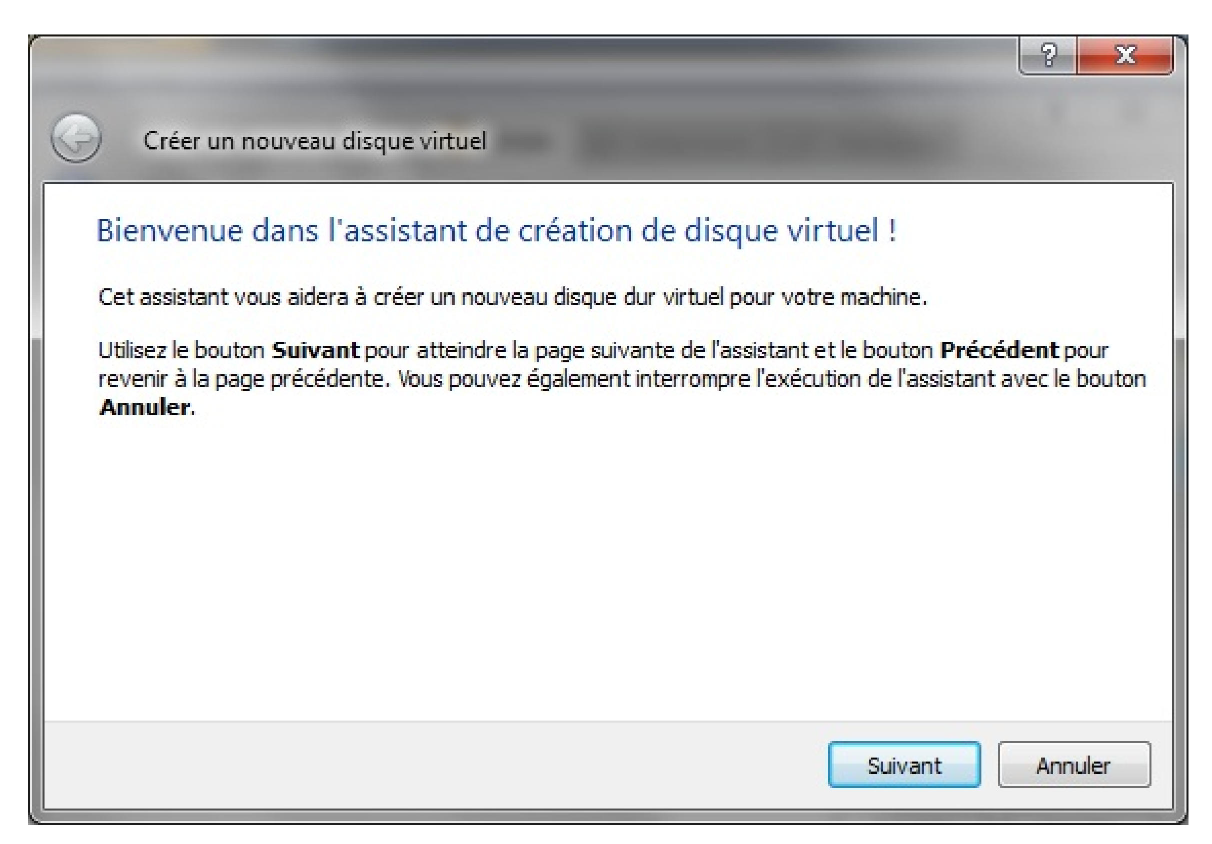

faire.

Vous devez ensuite indiquer quelle quantité de mémoire vive (RAM) vous souhaitez réserver à la machine virtuelle (figure suivante). En effet, pour que celle-ci fonctionne correctement, il va lui falloir de la mémoire… comme pour tout ordinateur normal ! Il va donc falloir « donner » un peu de mémoire à la machine : je vous recommande au moins 512 Mo, voire 1 Go si possible. VirtualBox peut occuper jusqu'à 50% de votre mémoire vive. Je dispose de 3 Go, la quantité maximale que l'on me propose est donc 1,5 Go.



Il nous reste maintenant à créer le disque dur de la machine virtuelle. VirtualBox va créer une sorte de gros fichier sur votre disque qui représentera le disque dur de la machine. Laissez l'option « Créer un nouveau disque dur » sélectionnée (figure suivante). Une nouvelle fenêtre s'ouvre alors : l'assistant de création de disque dur virtuel (figure suivante).



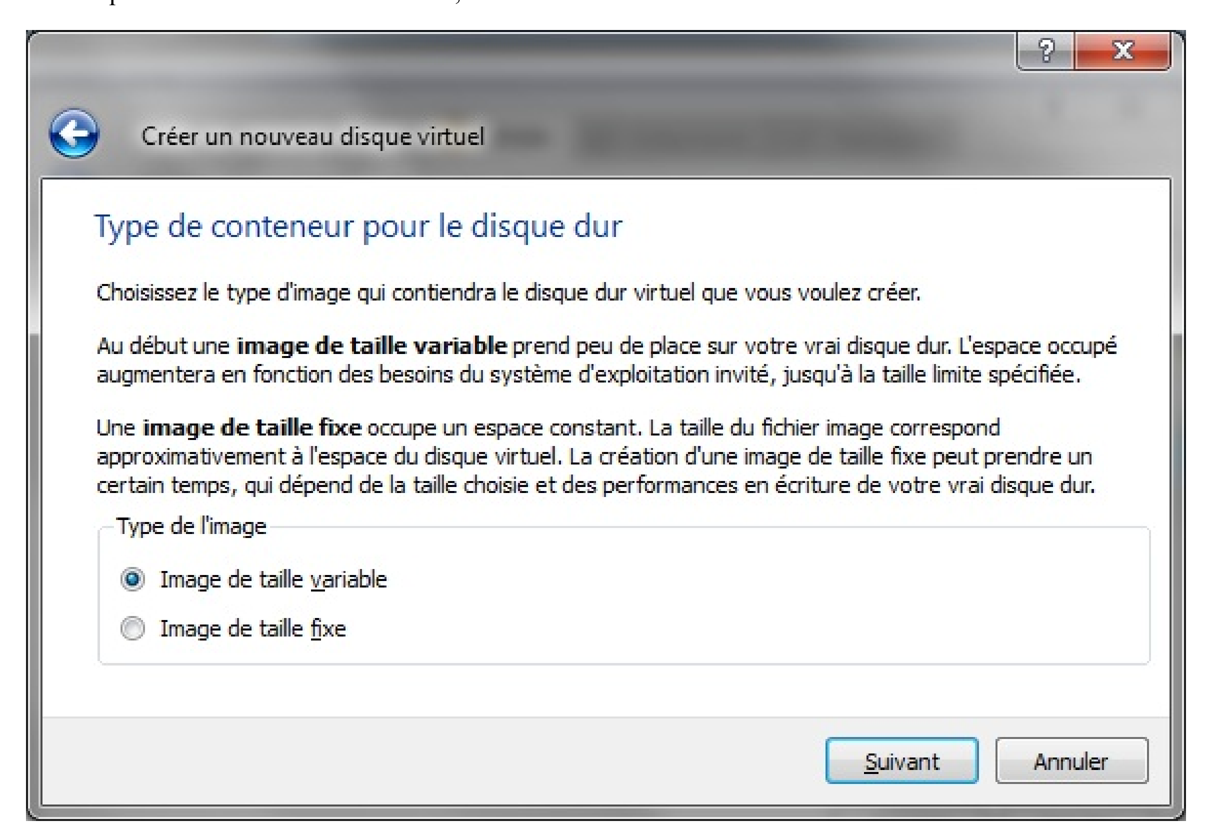


L'assistant de création de disque dur virtuel

L'assistant de création de disque dur virtuel vous demande quel type d'image disque vous souhaitez créer (figure suivante). Deux choix s'offrent à vous :

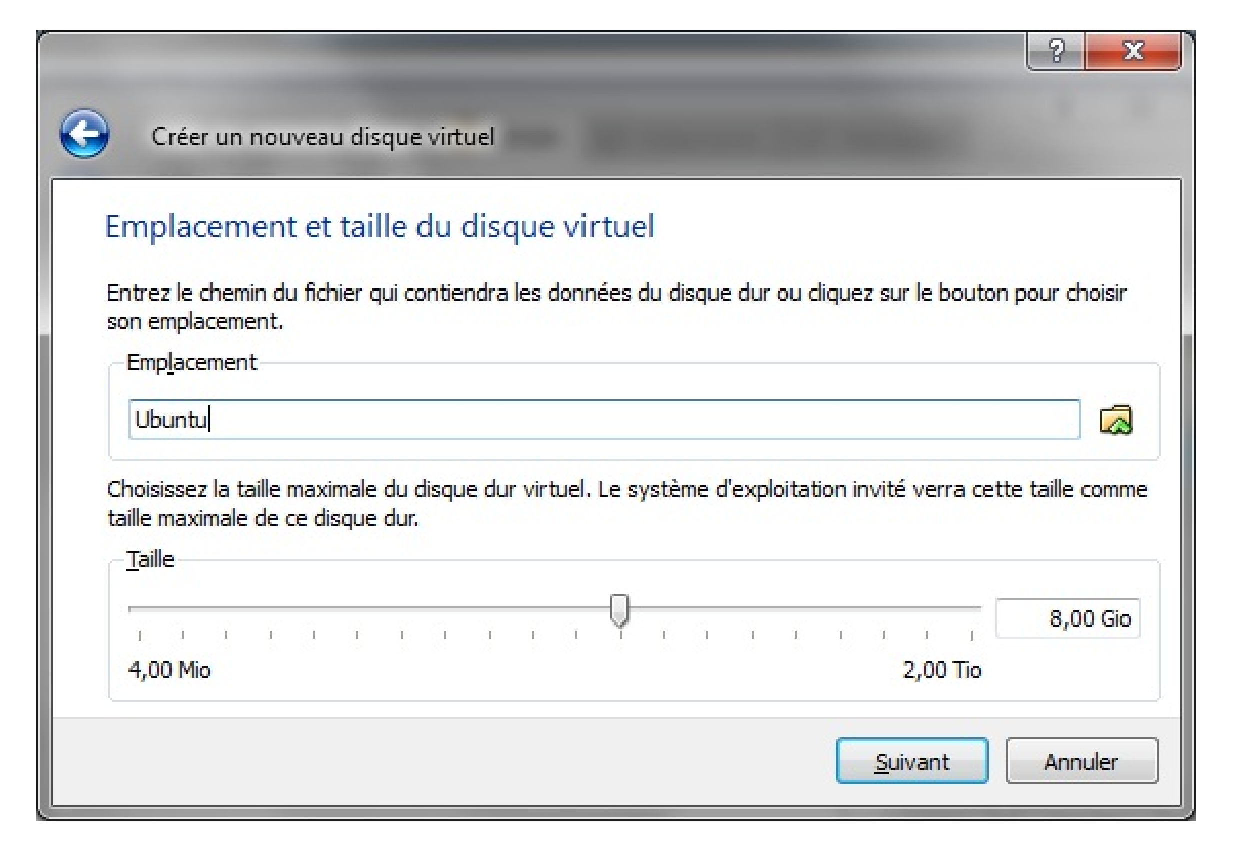
- **Image de taille variable** : le fichier « image » représentant le disque dur virtuel grossira en fonction de l'utilisation du disque dur. C'est l'option recommandée : si le disque virtuel a une taille totale de 8 Go et que seulement 2 Go sont utilisés, le fichier fera 2 Go.

- **Image de taille fixe** : le fichier « image » occupera immédiatement la place maximale. Si le disque virtuel a une taille totale de 8 Go et que seulement 2 Go sont utilisés, le fichier fera tout de même 8 Go.

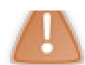


Je vous invite à choisir « Image de taille variable », sauf si vous ne manquez vraiment pas de place sur votre disque. 

Sur l'écran suivant (figure suivante), vous devrez donner un nom au disque dur virtuel ainsi qu'une taille maximale. Je vous recommande de laisser le nom par défaut (« Ubuntu ») et d'indiquer au moins 8 Go.



N'oubliez pas que la taille de l'image disque correspond à la taille maximale qui pourra être utilisée par la machine



virtuelle. Si vous avez l'intention d'installer de gros programmes, prévoyez un peu plus de place. Pour une installation classique et basique d'Ubuntu, 8 Go devraient cependant suffire.

Les assistants de création sont enfin terminés, ouf ! :) Ils ne sont pas si complexes en réalité ; il y a quelques années, la création de machine virtuelle était plutôt réservée aux experts. En quelques clics, nous avons configuré la machine, qui est maintenant prête à être lancée !

Lancer la machine virtuelle

L'écran d'accueil de VirtualBox devrait maintenant afficher une machine nommée « Ubuntu » dans la liste de gauche (figure suivante).

Oracle VM VirtualBox
Fichier Machine Aide
Nouvelle Configuration Démarrer Oublier
Ubuntu
Éteinte
Détails Instantanés Description

Général	
Nom :	Ubuntu
Type d'OS :	Ubuntu
Système	
Mémoire vive :	512 Mo
Nombre de processeurs :	1
Ordre d'amorçage :	Disquette, Disque CD/DVD, Disque dur
VT-x/AMD-V :	Activé
Pagination imbriquée :	Activée
Affichage	
Mémoire vidéo :	12 Mo
Accélération 3D :	Désactivée
Accélération 2D :	Désactivée
Serveur du bureau à distance (RDP) :	Désactivé
Stockage	
Contrôleur IDE	
Maître secondaire IDE (CD/DVD):	Vide
Contrôleur SATA	
Port SATA 0 :	Ubuntu.vdi (Normal, 8,00 Gio)
Son	

Configurer le lecteur CD

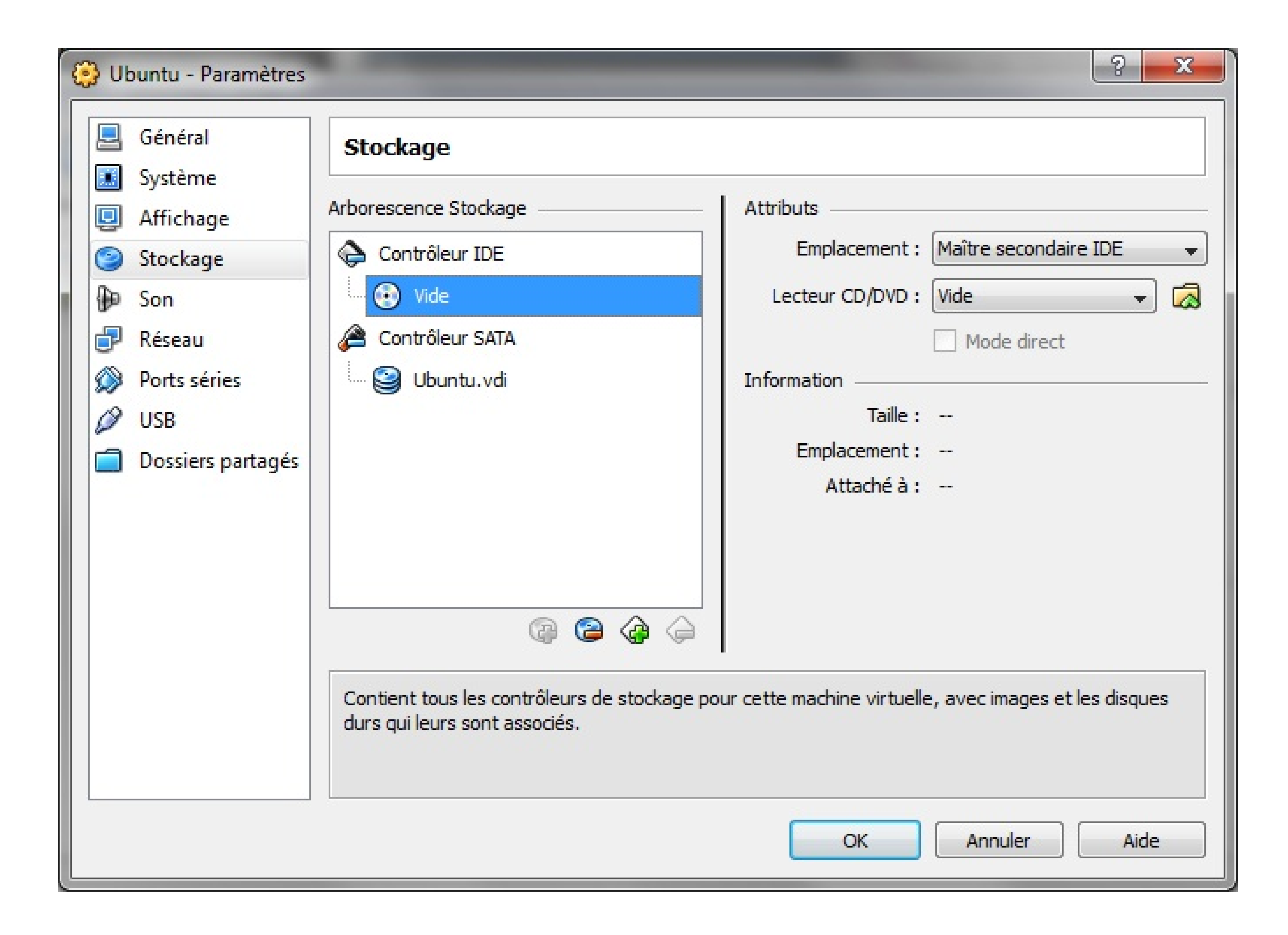
Avant de lancer la machine virtuelle, vous avez besoin du CD d'installation d'Ubuntu, exactement comme si vous démarriez votre ordinateur pour y installer Linux. Deux choix s'offrent à vous.

- Vous avez déjà gravé Ubuntu sur CD : il suffit d'insérer le CD dans le lecteur avant de lancer la machine virtuelle. Il s'agit du cas le plus simple.
- Vous avez téléchargé l'image (`.iso`) mais ne l'avez pas gravée sur CD : inutile d'utiliser un CD pour cela, VirtualBox est capable de lire directement l'image ISO.

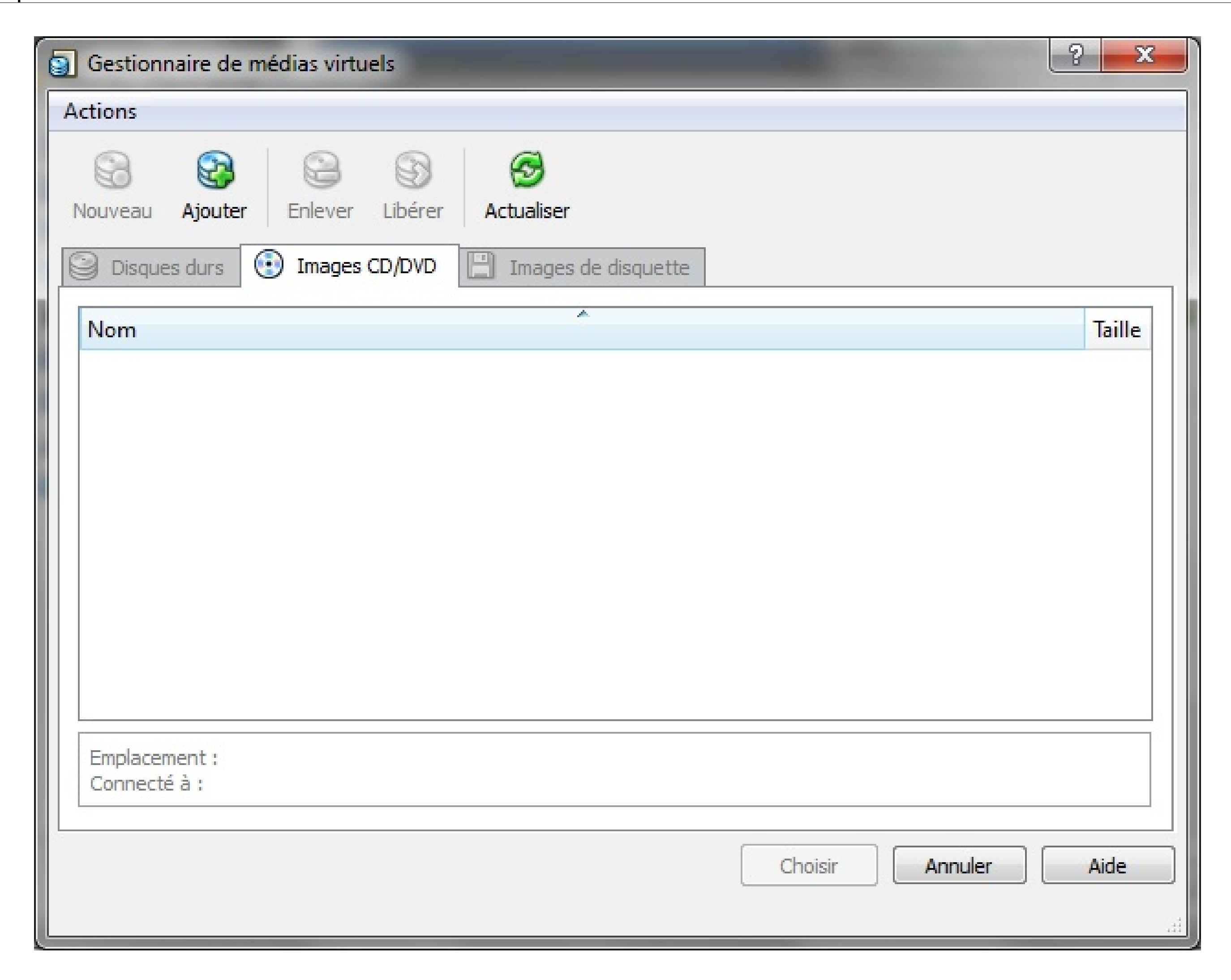
Dans le cas où vous avez l'image ISO sur votre disque dur et où vous ne souhaitez pas graver de CD, vous devez configurer la machine virtuelle pour qu'elle utilise le fichier `.iso` comme CD.

Sur l'écran d'accueil d'Ubuntu, cliquez tout d'abord sur le nom de la machine virtuelle disponible dans la liste puis cliquez sur le bouton « Configuration ».

Dans la fenêtre qui s'ouvre, sélectionnez « Stockage » dans la liste de gauche puis sélectionnez la ligne « Vide » sous « Contrôleur IDE » (figure suivante).



Cliquez sur la petite icône en forme de dossier à droite. Une nouvelle fenêtre s'ouvre : le gestionnaire de médias virtuels (figure suivante).



Ne vous laissez pas décourager par cette nouvelle fenêtre, nous avons bientôt terminé. Cliquez sur le bouton « Ajouter » et indiquez où se trouve l'image ISO que vous avez téléchargée sur le site d'Ubuntu.

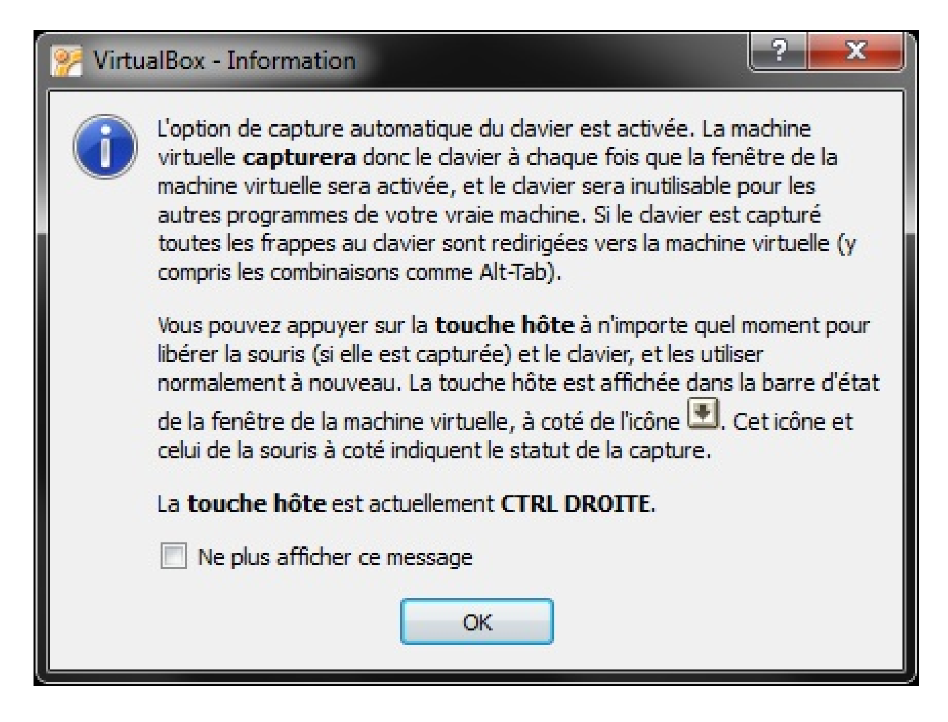
Il ne vous reste plus qu'à valider les fenêtres ; votre disque virtuel est prêt ! La machine peut enfin être lancée.

Premier démarrage de la machine

Pour lancer la machine, cliquez sur son nom dans la liste à gauche puis sur le bouton « Démarrer », en haut. Vous pouvez aussi double-cliquer sur le nom de la machine.

Une fenêtre représentant la machine virtuelle s'ouvre alors (figure suivante).

Au premier lancement, VirtualBox devrait vous afficher quelques messages d'information. Le plus important d'entre eux (figure suivante) vous indique qu'une touche spéciale (je l'appelle la « touche de secours ») vous permet de *sortir* de la machine virtuelle : il s'agit ici de `Ctrl Droite`, la touche `Ctrl` à droite du clavier.





Lorsque le curseur de la souris est à l'intérieur de la machine virtuelle, on dit que celle-ci « capture » la souris. Il en va de même pour le clavier. Il n'est pas possible d'en sortir pour revenir au système d'exploitation d'origine, à moins d'appuyer sur `Ctrl Droite`. Heureusement, depuis quelque temps, la capture de la souris se fait de façon plus transparente (on peut sortir facilement de la machine) et celle-ci est moins emprisonnée qu'elle ne l'était auparavant.

Ubuntu se charge sous vos yeux ébahis dans la machine virtuelle (figure suivante), et en moins de temps qu'il n'en faut pour le dire, vous voilà sur le bureau d'Ubuntu (figure suivante) !

À partir de là, vous pouvez tester Ubuntu et l'installer sans aucun risque.

Je vous invite donc à l'installer : tout se fera dans le disque virtuel (celui que nous avons créé tout à l'heure, qui est en fait un gros fichier sur votre disque dur). Vous verrez d'ailleurs à l'installation que vous disposez d'un disque dur de 8 Go, sous réserve que vous ayez défini un disque virtuel de cette taille.

Vous pouvez utiliser Ubuntu comme s'il était véritablement installé sur votre ordinateur ! Les performances sont légèrement moindres (car Windows tourne toujours en arrière-plan), mais cela est négligeable aujourd'hui, à moins que vous n'utilisiez des applications gourmandes comme les jeux.



Un des gros avantages de la machine virtuelle est qu'il est possible d'enregistrer son état à n'importe quel moment. C'est une sorte de sauvegarde instantanée. Pour y accéder, allez dans le menu `Machine → Prendre un instantané`. Vous pourrez par la suite revenir au moment exact de la sauvegarde.

Vous pouvez, si vous le désirez, afficher Ubuntu en plein écran. Il suffit d'effectuer la combinaison de touches `Ctrl Droite + F` (vous pouvez aussi aller dans le menu `Machine → Passer en plein écran`). Malheureusement, comme vous allez le constater, Ubuntu n'occupe pas tout votre écran. Pour y remédier, il va falloir installer les additions invité…

Installation des additions invité

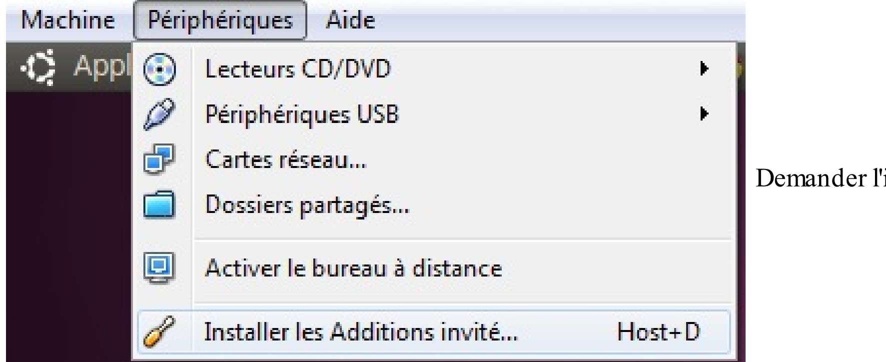
Nous avons installé Ubuntu dans une machine virtuelle et cela fonctionne déjà très bien. Néanmoins, il est recommandé d'y installer ce que l'on appelle les *additions invité*. Ce sont en fait des pilotes spéciaux que l'on installe dans la machine virtuelle pour améliorer ses performances.

Ubuntu n'a pour le moment pas « conscience » qu'il s'agit d'une machine virtuelle. Il s'exécute comme il le ferait sur une vraie machine. Le rôle des additions est de modifier légèrement Ubuntu pour qu'il prenne conscience qu'il est dans une machine virtuelle, ce qui aura pour effet d'améliorer son fonctionnement.

Parmi les améliorations apportées par cette modification, on note :

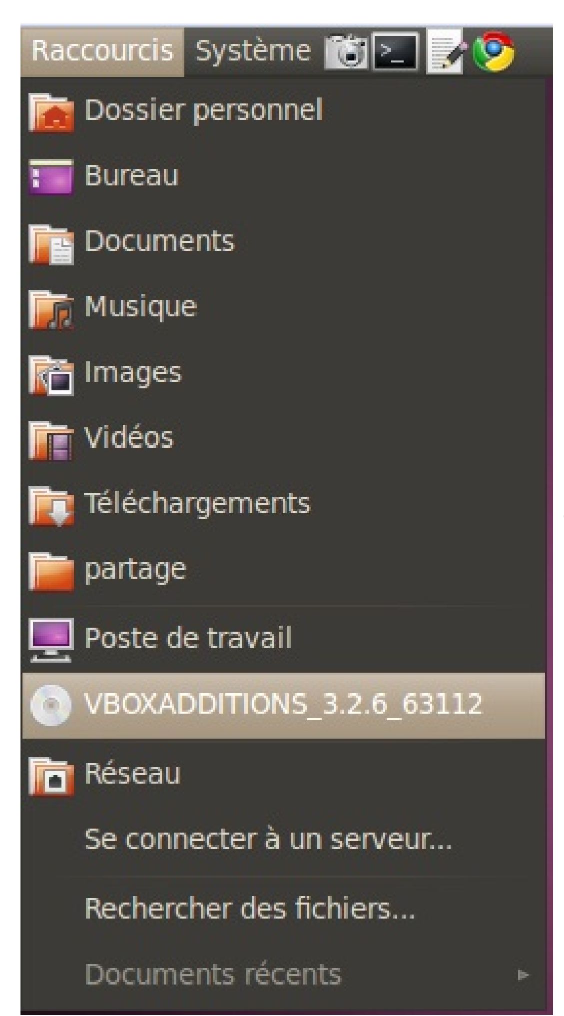
- de meilleures **performances graphiques** : le déplacement des fenêtres et les animations seront plus fluides ;
- un meilleur **suivi de la souris** : la souris réagira de façon plus naturelle ;
- un accès à de plus **grandes résolutions**, capables de suivre en temps réel la taille de la fenêtre de VirtualBox si vous la redimensionnez ou si vous l'affichez en plein écran (avec `Ctrl Droite + F`) ;
- le **partage du presse-papier** entre Windows et Linux. Vous pourrez copier du texte dans Linux et le coller dans Windows, et vice-versa !
- les **répertoires partagés** : vous pouvez faire en sorte qu'un répertoire de Windows apparaisse aussi sous Linux dans la machine virtuelle. Tous les changements dans ce dossier seront immédiatement répercutés sur les deux systèmes.

Pour installer les additions, rendez-vous dans le menu `Périphériques → Installer les Additions invité`. Vous pouvez aussi faire la combinaison de touches `Ctrl Droite + D` (figure suivante).



Demander l'installation des additions

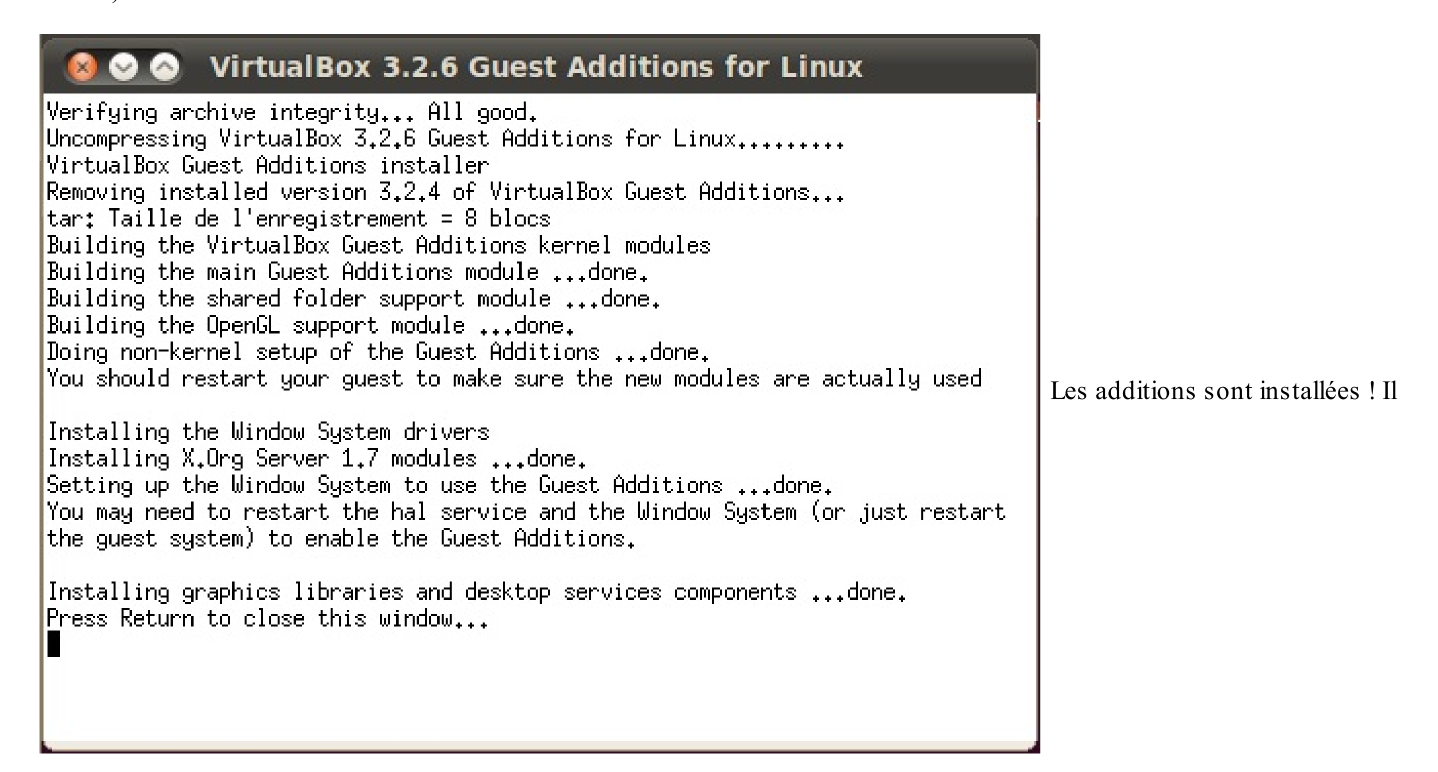
Cela provoque l'insertion d'un CD virtuel dans la machine. Vous pouvez accéder au contenu du CD en ouvrant le menu `Raccourcis` d'Ubuntu (figure suivante). Une fenêtre affichant les fichiers du CD apparaît : double-cliquez sur `autorun.sh` (figure suivante). Cliquez sur « Lancer dans un terminal » lorsqu'on vous demande ce que vous souhaitez faire.



Ouvrez le CD des additions qui vient d'apparaître

Lancez le programme autorun.sh

Vous devrez à nouveau indiquer votre mot de passe par mesure de sécurité. Une console s'ouvre et les additions s'installent dans Ubuntu. Patientez jusqu'à ce que l'on vous demande d'appuyer sur `Entrée` (« Return » en anglais) pour fermer la fenêtre (figure suivante).



Les additions sont installées ! Il

faut alors appuyer sur Entrée et redémarrer.

Les additions sont installées ! Pour qu'elles soient prises en compte, vous devrez ensuite redémarrer Ubuntu.



Vous pouvez maintenant partager un dossier entre Windows et Ubuntu : rendez-vous dans le menu `Périphériques → Dossiers partagés`. Vous pourrez y indiquer un dossier existant de Windows et le nom du dossier équivalent dans Ubuntu.

En résumé

- VirtualBox est un outil qui permet de faire tourner un ordinateur virtuel au sein d'un système d'exploitation.
- Nous utilisons ici VirtualBox pour installer Linux à l'intérieur de Windows. Il n'y a aucun risque de conflit entre les deux car la machine virtuelle est cloisonnée.
- Vous pouvez utiliser la machine virtuelle comme un véritable ordinateur mais vous perdez légèrement en performances. Cette technique est à réserver à ceux qui souhaitent ne prendre aucun risque lors de l'installation d'Ubuntu ou qui désirent simplement essayer la distribution.
- Une fois Ubuntu installé dans la machine virtuelle, il est conseillé d'installer les additions invité. Cela améliorera les performances et vous permettra notamment d'utiliser Ubuntu en plein écran, de partager des dossiers et le presse-papier, etc.

Partie 2 : Manipuler la console et les fichiers

 La console, ça se mange ?

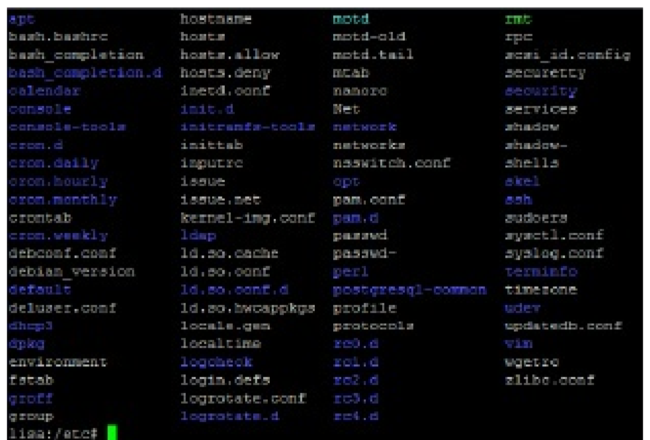
Tout au long de la première partie du livre, nous avons passé notre temps à découvrir en douceur Linux, son bureau et son interface graphique. Par rapport à d'autres systèmes d'exploitation comme Windows, c'est un peu dépaysant au début, mais mine de rien, on retrouve beaucoup de concepts similaires. Je pense donc que vous n'aurez plus besoin de moi pour être capables de manipuler correctement l'interface graphique.

Les choses intéressantes commencent **maintenant**. C'est à partir d'ici qu'un utilisateur classique de Windows met les pieds dans un environnement totalement nouveau. Vous ne pouvez pas avoir d'a priori, et il y a de fortes chances que ce soit un domaine de l'informatique que vous n'ayez jamais approché (non, non, DOS ne compte pas).

Vous n'avez pas idée de la richesse quasi-infinie offerte par la console. Personne ne peut d'ailleurs prétendre la maîtriser entièrement, c'est vous dire ! Vous aurez donc toujours quelque chose à découvrir. :-)

Pourquoi avoir inventé la console ?

Avant de vous lancer à corps perdu dans l'océan de la console, ce chapitre va vous enseigner les rudiments de survie pour éviter la noyade. Parce que bon, ce serait dommage que vous vous arrêtiez avant le meilleur moment.



Une console

On va commencer par répondre à cette question hautement fondamentale :



Mais pourquoi ont-ils inventé la console au lieu de l'interface graphique, d'abord ? C'est quand même plus pratique une interface graphique avec une souris ; c'est plus intuitif ! C'est juste pour faire pro, faire compliqué pour faire compliqué et s'assurer que l'informatique reste seulement à la portée de quelques initiés ? Pourquoi ne pas avoir supprimé la console ? C'est archaïque ! (ce sont les questions que vous devez vous poser, je me trompe ?)

Que nenni ! Il y a une explication à tout ; voici une réponse point par point.

Pourquoi avoir inventé la console d'abord, au lieu de l'interface graphique ?

Pour ça, je vous ai mis la puce à l'oreille dès le premier chapitre. La réponse est : **parce qu'on n'avait de toute façon pas le choix** !
Les débuts de l'informatique et de la console remontent aux débuts des années 70, à une époque où un écran 2 couleurs était un luxe inimaginable et où la puissance de calcul de ces ordinateurs était cent fois plus faible que celle de la calculatrice Casio de ma petite sœur. Bref, vous voyez le genre.
On ne dirait pas comme ça, mais gérer une interface graphique avec plusieurs couleurs ainsi qu'une souris et un certain nombre de fonctionnalités avancées qui vous paraissent aujourd'hui « normales », ça demande de la puissance ! La console était donc à cette époque la seule façon d'utiliser un ordinateur.

L'interface graphique avec la souris, c'est quand même plus intuitif !

Alors là, tout à fait d'accord avec vous. On dira ce qu'on voudra, mais la console n'est PAS intuitive. Quand on débute en informatique, il est de loin plus simple d'appréhender l'interface graphique. En revanche, je suis aujourd'hui persuadé que l'interface graphique de Linux (que ce soit KDE, Unity ou une autre) est aussi intuitive que celle de Windows et de Mac OS. Ça n'a pas toujours été forcément le cas, mais un débutant total en informatique n'aura pas plus de mal à appréhender l'interface graphique de Linux que celle de Windows ; ça, j'en suis totalement convaincu.

Est-ce que c'est juste pour faire pro, inutilement compliqué ?

Les commandes de la console vont peut-être vous sembler être du chinois les premiers temps, et vous allez vous demander à coup sûr si cet amas de lettres vide de sens n'est pas là juste pour faire en sorte que le moins de monde possible puisse utiliser la console (sous-entendu : « Seuls les programmeurs qui ont inventé la console devraient pouvoir l'utiliser. »). Ça, par contre, c'est totalement faux. Tout a été minutieusement pensé, et ce dès les années 60.

- **Les commandes sont courtes, abrégées.** C'est pour gagner du temps et aller plus vite. Écrire `pwd` est moins intuitif que `diredansquelrepertoirejesuis`, mais après l'avoir écrit deux cents fois dans la journée, vous bénirez les programmeurs qui ont fait ce choix, croyez-moi !
- **Les commandes ne sont pas intuitives.** Faux. Il s'agit bien souvent d'une abréviation de termes (en anglais, *of course* !) et les lettres qu'il faut taper sont généralement choisies en fonction de leur proximité les unes par rapport aux autres pour que vous ayez le moins possible à déplacer les doigts sur le clavier ! Bon, d'accord : à la base, c'est plutôt fait pour les claviers QWERTY anglais qui sont – je le reconnais – plus adaptés pour accéder aux symboles du genre `{ ] | #`, etc. Mais vous n'en mourrez pas. ;-)

Pourquoi ne pas avoir supprimé la console ? C'est archaïque !

Depuis l'invention de l'interface graphique, on pourrait se demander pourquoi on n'a pas supprimé la console (sous-entendu : « Elle ne sert plus à rien »). C'est là que beaucoup se trompent complètement : on met un peu de temps à s'y faire, mais quand on sait s'en servir, on va **beaucoup** plus vite avec la console qu'avec l'interface graphique. C'est même pire en fait : vous vous rendrez compte à un moment qu'il y a des choses que seule la console peut faire et qu'il serait de toute façon vraiment inutile de recourir à une interface graphique pour les effectuer.

Un exemple ? En mode graphique, allez dans un répertoire qui contient beaucoup de fichiers en tout genre : des fichiers texte, des images, des vidéos… Vous voudriez savoir combien il y a d'images JPEG dans ce dossier : pas facile hein ? :-D
En console, en assemblant quelques commandes, on peut obtenir ce résultat sans problème !

Code : Console

```
ls -l | grep jpg | wc -l
510
```

La première ligne est la commande que j'ai tapée, la seconde le résultat. Il y avait donc 510 images JPEG dans le dossier, et on a obtenu le résultat en moins d'une seconde !
On peut même faire encore plus fort et enregistrer directement ce nombre dans un fichier texte :

Code : Console

```
ls -l | grep jpg | wc -l > nb_jpg.txt
```

… et on peut aussi envoyer le fichier `nb_jpg.txt` sur Internet par FTP ou à un ami par e-mail, le tout en une ligne !
La console n'est donc pas morte et n'a pas du tout prévu de l'être !

La plupart des commandes de la console de Linux sont des « copies » d'Unix, ce vieil OS dont je vous ai parlé au tout début,

ancêtre parmi les ancêtres. N'allez pas croire que les programmes d'Unix ont été copiés ou « piratés » par Linux ; c'est juste que leur mode d'emploi est le même. Les programmes ont été réécrits par un groupement de programmeurs issus de ce qu'on appelle le projet GNU.
Ce projet a fusionné au bout de quelque temps avec le cœur du système d'exploitation Linux pour donner au final GNU/Linux, qu'on écrit en pratique juste « Linux » car c'est plus court. Mais tout ça, je vous l'ai déjà dit dans le premier chapitre.

L'avantage ? Les commandes n'ont pas bougé et ne bougent pas depuis l'époque d'Unix (soit depuis les années 60). Ce sont les mêmes. Quelqu'un qui utilisait Unix dans les années 60 est capable de se débrouiller avec un Linux d'aujourd'hui. Et il y a fort à parier que ce sera pareil pour les nombreuses années à venir. Vous avez donc juste à apprendre à vous en servir une fois. O.K., il y aura du boulot, mais après ce sera quelque chose qui pourra vous servir toute votre vie !



Ce que vous apprendrez dans les chapitres suivants sera l'utilisation de commandes de type Unix. L'avantage, c'est que ça ne marchera pas seulement sous Linux mais aussi sous tous les OS également basés sur Unix !
En effet, à quelques exceptions près, tout ce que vous verrez pourra donc être fait sous les OS basés sur Unix, en particulier Mac OS X.
Si vous avez Mac OS X et que vous souhaitez ouvrir une console, faites `Pomme + Shift + U` pour ouvrir les Utilitaires, et sélectionnez « Terminal » comme le montre la capture d'écran de la figure suivante



Ouvrir une console sous MAC OS X

La console, la vraie, celle qui fait peur

Bien : il s'agit maintenant de savoir comment approcher la Bête. La question est donc : comment accède-t-on à la console dans son beau Linux depuis son KDE / Unity / XFCE / *insérez le nom de votre gestionnaire de bureau ici* ?

Les moyens sont variés, très variés. Il y a donc le choix, un peu comme partout sous Linux me direz-vous.

Nous allons commencer par la solution la plus « basique » et que vous utiliserez probablement le moins souvent. Elle vous permet d'accéder à la vraie console (si tant est qu'il y ait une « vraie » console) en pressant une combinaison de touches.



Lisez bien tous les paragraphes qui suivent avant d'exécuter les commandes que je vais vous donner. Ce n'est pas dangereux rassurez-vous, mais c'est juste que si vous vous retrouvez en console avant d'avoir lu comment en sortir, vous aurez l'air bien embêtés. ;-)

Je vais supposer que vous vous êtes connectés, c'est-à-dire que vous avez entré votre login et votre mot de passe. Vous êtes donc sur votre gestionnaire de bureau, ici Unity (figure suivante).

Gestionnaire de bureau Unity

Vous trouvez qu'il y a trop de couleurs ? Que ça manque de mots compliqués ?
Pas de problème ! Voici les raccourcis à connaître pour accéder à la console :

- `Ctrl + Alt + F1` : terminal 1 (tty1) ;
- `Ctrl + Alt + F2` : terminal 2 (tty2) ;
- `Ctrl + Alt + F3` : terminal 3 (tty3) ;
- `Ctrl + Alt + F4` : terminal 4 (tty4) ;
- `Ctrl + Alt + F5` : terminal 5 (tty5) ;
- `Ctrl + Alt + F6` : terminal 6 (tty6) ;
- `Ctrl + Alt + F7` : retour au mode graphique (ouf !).



« Terminal » est un autre nom pour « Console ».



Attention : pensez bien, si vous testez, que vous serez alors en mode console. Vous devrez donc utiliser `Ctrl + Alt + F7` pour revenir en mode graphique. N'oubliez pas !

Pour tester, tapez `Ctrl + Alt + F1`.
Votre écran va peut-être clignoter quelques instants ; ne paniquez pas. Vous allez ensuite entrer en mode console plein écran (figure suivante).

```
Starting up ...
Loading, please wait...
kinit: name_to_dev_t(/dev/disk/by-uuid/890a63fb-b2e0-4e1f-841c-b176acae2e27) = s
da5(8,5)
kinit: trying to resume from /dev/disk/by-uuid/890a63fb-b2e0-4e1f-841c-b176acae2
e27
kinit: No resume image, doing normal boot...

Ubuntu 7.04 mateo21-desktop tty1

mateo21-desktop login:
```

Mode console plein écran – C'est beau, hein ?

Le login

Vous ne pouvez pas utiliser tout de suite la console : il faut d'abord vous logger. C'est ce que vous demande la dernière ligne :

Code : Console

```
mateo21-desktop login:
```

`mateo21-desktop` : c'est le nom que vous avez donné à votre ordinateur lors de l'installation. Votre ordinateur se présente et vous rappelle où vous êtes, en quelque sorte.
Ça a l'air inutile comme ça, mais avec Linux on peut se connecter à un autre PC facilement en console (on en parlera plus loin), et parfois on ne sait plus si on est dans la console de son PC ou dans celle d'un autre.

Bon, entrez votre login ; dans mon cas c'est `mateo21`.

On vous demande ensuite votre mot de passe :

Code : Console

```
Password:
```

Là, vous rentrez votre mot de passe. Dans mon cas c'est ~~ooops~~ euh… j'ai rien dit.

Ne soyez pas étonnés si les lettres que vous tapez n'apparaissent pas. En fait, il n'y a même pas d'étoiles qui s'affichent à l'écran. Cela permet d'éviter qu'une personne derrière vous compte le nombre de caractères de votre mot de passe (euh oui, sous Linux on est un peu parano).

Si tout va bien, vous devriez voir l'écran de la figure suivante.

```
Starting up ...
Loading, please wait...
kinit: name_to_dev_t(/dev/disk/by-uuid/890a63fb-b2e0-4e1f-841c-b176acae2e27) = s
da5(8,5)
kinit: trying to resume from /dev/disk/by-uuid/890a63fb-b2e0-4e1f-841c-b176acae2
e27
kinit: No resume image, doing normal boot...

Ubuntu 7.04 mateo21-desktop tty1

mateo21-desktop login: mateo21
Password:
Last login: Sat Aug 18 17:44:52 2007 on :0
Linux mateo21-desktop 2.6.20-16-generic #2 SMP Thu Jun 7 20:19:32 UTC 2007 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
mateo21@mateo21-desktop:~$ _
```

Mode console après connexion

La console devrait afficher en bas une ligne similaire à celle-ci :

Code : Console

```
mateo21@mateo21-desktop:~$ _
```

C'est bon, vous y êtes. :-)

Les différentes consoles

Sous toute machine Linux, il y a donc non pas une mais six consoles qui fonctionnent en simultané (d'où les six raccourcis différents de `Ctrl + Alt + F1` à `Ctrl + Alt + F6`).

Vous pouvez savoir dans quel terminal vous êtes lors du chargement : il est en effet marqué « tty1 » si vous êtes sur le terminal n°1. Regardez de plus près mon image (figure suivante).

```
Starting up ...
Loading, please wait...
kinit: name_to_dev_t(/dev/disk/by-uuid/890a63fb-b2e0-4e1f-841c-b176acae2e27) = s
da5(8,5)
kinit: trying to resume from /dev/disk/by-uuid/890a63fb-b2e0-4e1f-841c-b176acae2
e27
kinit: No resume image, doing normal boot...

Ubuntu 7.04 mateo21-desktop tty1

mateo21-desktop login: mateo21
Password:
Last login: Sat Aug 18 17:44:52 2007 on :0
Linux mateo21-desktop 2.6.20-16-generic #2 SMP Thu Jun 7 20:19:32 UTC 2007 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
mateo21@mateo21-desktop:~$ _
```

Console « tty1 »

L'information est un peu cachée mais elle est là.
Au pire, vous changez de terminal jusqu'à retrouver celui sur lequel vous êtes ; dès que vous en avez marre, vous pouvez retourner au mode graphique avec `Ctrl + Alt + F7`.



À noter qu'une combinaison similaire permet de redémarrer le serveur X, c'est-à-dire l'interface graphique. Contrairement aux apparences, ça ne redémarre pas l'ordinateur mais juste l'élément graphique. Les consoles continuent à tourner derrière.
Cette combinaison spéciale, c'est `Ctrl + Alt + Backspace` (la touche « Retour arrière »). Attention, c'est assez radical. Pensez donc à enregistrer vos documents avant d'essayer. Vous ne devriez pas avoir à le faire souvent, sauf si l'interface graphique est plantée (rare) ou si on vous demande de le faire pour prendre en compte une nouvelle configuration de X. Je vous dirai ça en temps voulu.
Notez enfin que sous les dernières versions d'Ubuntu, ce raccourci pourtant commun a été remplacé par `Alt + Impr.Écran + K`. La combinaison `Ctrl + Alt + Backspace` a été considérée comme trop fréquemment utilisée de manière inappropriée par les débutants habitués à une combinaison similaire présente sous Windows.

La console en mode graphique

Bon ! La console en plein écran, c'est bien joli mais pas franchement folichon. Je voulais vous montrer ça parce que c'est la vieille technique qui marche partout, même quand il n'y a pas d'interface graphique installée (c'est le cas sur la plupart des serveurs tournant sous Linux, par exemple).

Mais dans le cas qui nous intéresse, c'est-à-dire le vôtre, il y a bien mieux, croyez-moi. Même les plus fous de Linux préfèrent ouvrir une console dans le mode graphique. Les raisons sont multiples :

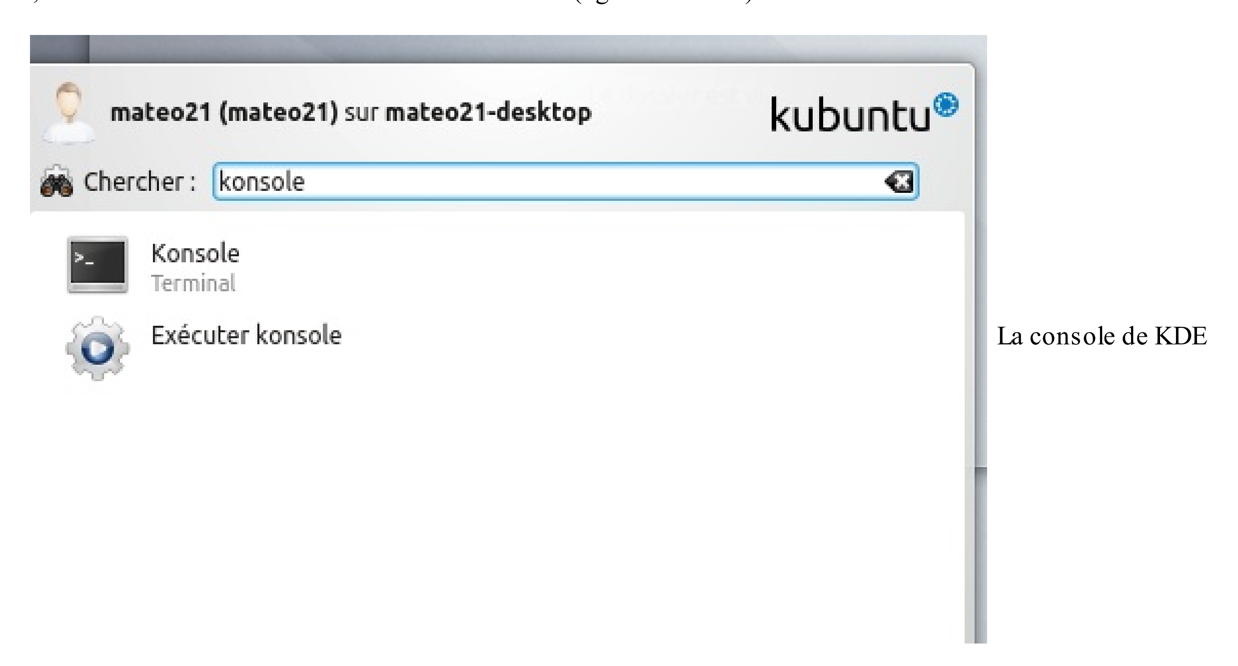
- déjà parce que c'est plus sympa et que vous pouvez en même temps continuer à utiliser d'autres applications graphiques tout en discutant avec vos amis en ligne… ;
- mais aussi parce que la résolution de l'écran est plus grande et qu'on peut afficher plus de choses à la fois dans la console… ;
- parce qu'on peut personnaliser l'apparence de la console et mettre, pourquoi pas, une image de fond… ;
- et qu'on peut aussi utiliser la souris pour copier-coller du texte dans la console (comme quoi, la souris sert quelques rares fois en console !).

Je vous conseille donc vivement d'utiliser autant que possible cette console en mode graphique, notamment tout au long de la lecture de ce livre. C'est juste plus confortable.

Comment accéder à la console en mode graphique ? Tout dépend de votre gestionnaire de bureau. Chacun propose un programme de console différent (mais tous se valent, globalement).

Sous KDE

Pour KDE, il suffit de rechercher "Konsole" dans le lanceur (figure suivante).



La console de KDE

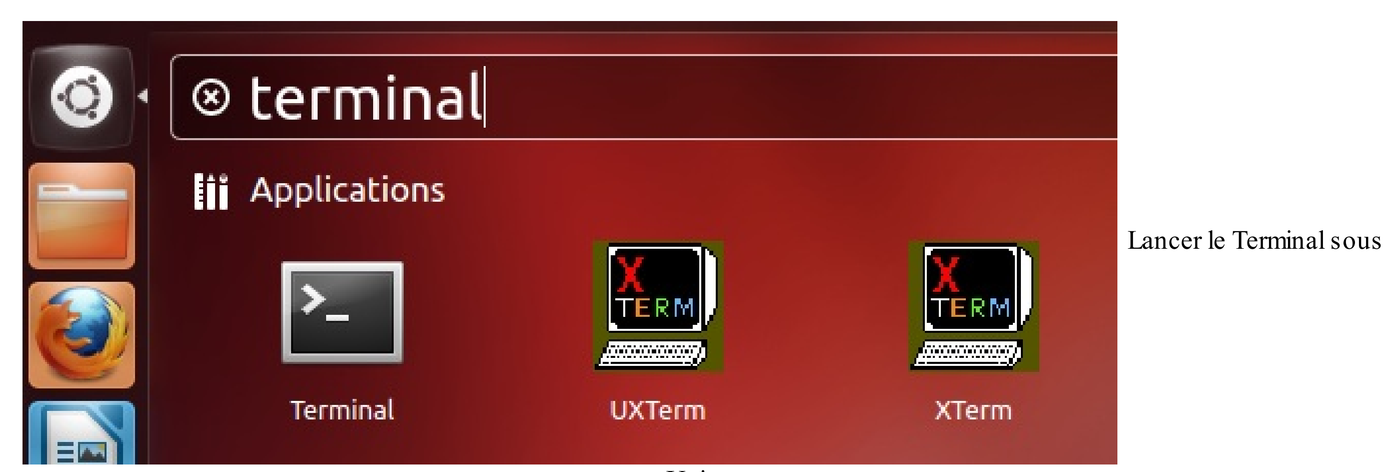
La console de KDE est visible sur la figure suivante.

Console de KDE

Vous pouvez vous amuser à changer la couleur de fond ou à insérer une image à la place : vous trouverez ça dans les menus.

Sous Unity

Si vous êtes sous Unity, rendez-vous dans le lanceur et recherchez "Terminal" (figure suivante).



Lancer le Terminal sous Unity

Voyez la console de Unity sur la figure suivante.

Terminal de Unity

Vous noterez que les deux types de console permettent d'ouvrir plusieurs onglets. Pour celle de KDE, il y a une petite icône en bas à gauche, et pour celle de Unity il y a le menu `Fichier / Ouvrir un onglet`.
Les onglets sont en général très pratiques car ils permettent de multiplier les consoles et donc de faire plusieurs choses en même temps.

L'accès à distance en SSH avec PuTTY

Cette partie sur l'accès à distance en SSH avec PuTTY ne sert qu'à vous montrer les possibilités d'utilisation de la console. **N'essayez pas de faire ça pour le moment** car il y a des détails un peu compliqués.
En clair : lisez ce que j'ai à vous dire ; c'est pour votre culture, pour que vous sachiez que ça existe. On verra SSH en pratique un peu plus tard dans le livre.

Une des grosses forces de la console, c'est d'être accessible à distance par Internet. Il suffit que votre machine soit connectée au réseau pour que vous puissiez vous logger de n'importe quel ordinateur dans le monde et faire comme si vous étiez chez vous ! Ça peut être pratique pour une foule de choses, comme surveiller l'état d'un téléchargement un peu long, lancer l'exécution d'un programme pour qu'il soit prêt lorsque vous serez rentrés chez vous… mais surtout, c'est comme ça que l'on administre un serveur sous Linux.

Un serveur est – pour faire simple – un ordinateur tout le temps connecté à Internet. Il permet d'offrir des services divers et variés aux internautes. Par exemple, il y a des serveurs web dont le rôle est de… distribuer des pages web.
La grande majorité des serveurs tourne sous Linux. Lorsque vous allez sur un site, il y a de très fortes chances pour que ce soit un serveur Linux qui vous réponde. Les serveurs Windows existent aussi, mais ils sont plus rares et on apprécie en général la stabilité de Linux ainsi que la possibilité de l'administrer à distance en ligne de commande.

On aura l'occasion de reparler de serveurs Linux plus tard (c'est un vaste sujet), mais je souhaite déjà vous montrer rapidement comment on fait pour se connecter à distance.

Telnet et SSH

Pour communiquer entre votre ordinateur et le serveur, il faut un **protocole**. C'est un ensemble de règles pour que deux ordinateurs puissent discuter entre eux… un peu comme si deux personnes devaient parler la même langue pour avoir une conversation.

Il existe des tonnes et des tonnes de protocoles pour communiquer par Internet, mais pour ce qui est d'accéder à la ligne de commande à distance, c'est-à-dire à la console, il y en a deux principaux.

- **Telnet** : le protocole le plus basique, qui présente le gros défaut de ne pas crypter les données échangées entre vous et le serveur. Si un pirate « écoute » vos échanges par un moyen ou un autre, il pourrait récupérer des informations sensibles, en particulier votre mot de passe lorsque vous l'envoyez à la connexion. Ce moyen de connexion reste utilisé mais peu par rapport à SSH.
- **SSH** : c'est de très loin le protocole le plus utilisé (et que l'on préfère) car il permet de crypter les données et de sécuriser ainsi la connexion avec le serveur.

Vous l'aurez compris, vous entendrez donc davantage parler de SSH que d'autre chose.

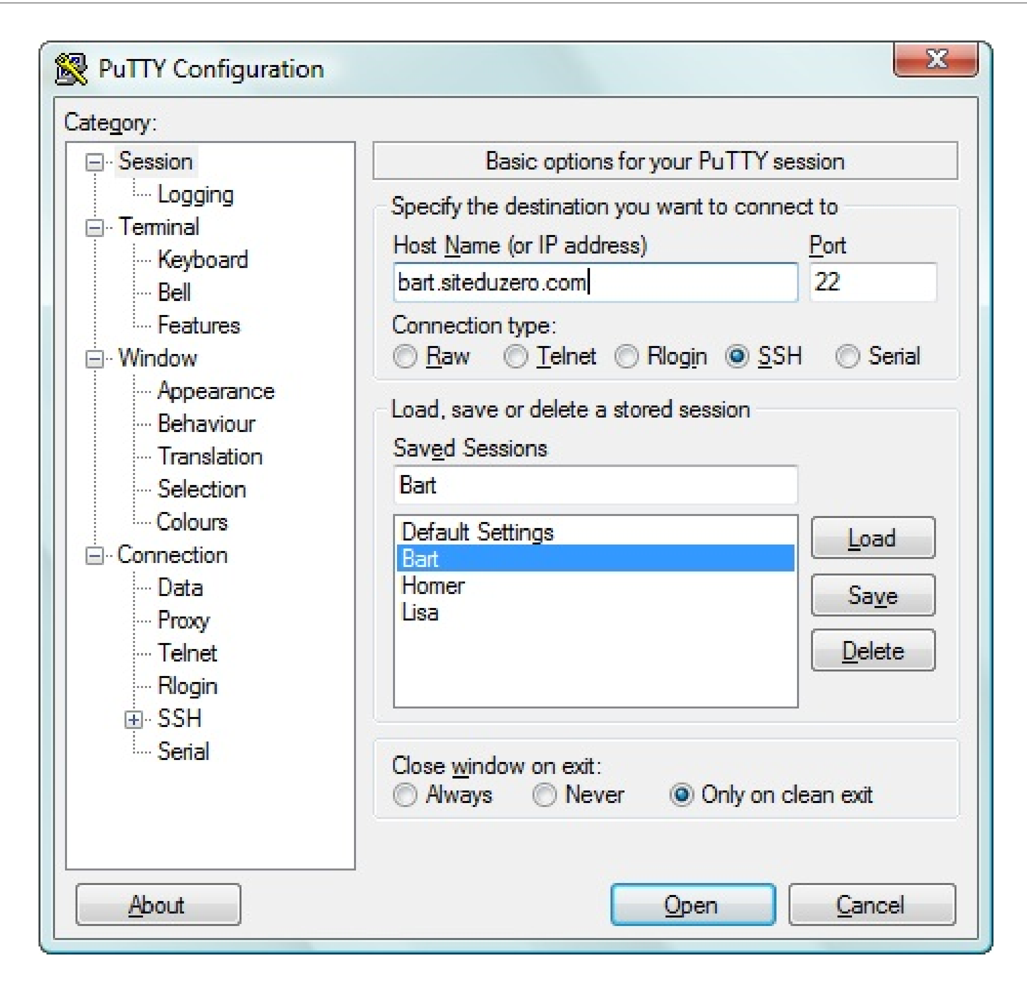
PuTTY

Pour accéder à distance à un ordinateur sous Linux connecté à Internet, vous avez besoin d'un programme spécial capable de restituer la ligne de commande à distance. Ce qui est bien, c'est que vous n'avez pas forcément besoin d'être sous Linux pour vous connecter à un autre ordinateur utilisant cet OS ; on peut très bien le faire depuis Windows, et c'est d'ailleurs la procédure que je vais vous montrer ici.

Il existe plusieurs programmes capables de se connecter en SSH à un serveur Linux. Le plus célèbre sous Windows est sûrement PuTTY : il est gratuit, léger et ne nécessite même pas d'installation (juste un exécutable à lancer).

Pour le télécharger, allez sur le site web du logiciel. Rendez-vous sur la page « Download » et cliquez sur « putty.exe ».

Lorsque vous le lancez, la fenêtre de configuration s'affiche (figure suivante).



Il y a beaucoup de pages d'options, comme le montre la section à gauche de la fenêtre. Dans la majeure partie des cas, vous n'aurez pas besoin d'y aller, heureusement.
Seule la première page est en fait vraiment importante : vous devez indiquer en haut dans le champ `Host Name` le nom d'hôte du serveur (dans mon cas `bart.siteduzero.com`) ou encore l'adresse IP de l'ordinateur, ce qui marche aussi bien (c'est juste plus difficile à retenir). Vérifiez que le type de connexion sélectionné en dessous est bien `SSH`, puis cliquez sur le bouton `Open` tout en bas.



Si vous avez l'habitude de vous connecter à plusieurs serveurs différents, sachez qu'il est possible de sauvegarder les IP et configurations pour se connecter à chacun d'eux. Utilisez la section « `Saved sessions` » au centre pour enregistrer ou ouvrir des connexions pré-enregistrées.

Après avoir cliqué sur `Open`, une fenêtre vous demandera lors de la première connexion au serveur si vous voulez stocker l'empreinte de ce dernier. C'est une sécurité pour vérifier que le serveur n'a pas changé depuis la dernière connexion et donc pour éviter que quelqu'un se fasse passer pour le serveur auquel vous avez l'habitude de vous connecter (le monde des pirates est sans pitié !).
Par la suite, on ne vous embêtera normalement plus jamais avec cette fenêtre.

La fenêtre principale de PuTTY s'affiche alors (figure suivante).

Voilà, vous n'avez plus qu'à vous connecter.
Indiquez votre login (par exemple `mateo21`), puis tapez `Entrée`.
On vous demande ensuite votre mot de passe ; tapez-le puis faites `Entrée`.



Encore une fois, il est normal de ne pas voir d'étoiles lorsque vous tapez votre mot de passe. C'est une sécurité supplémentaire pour éviter que quelqu'un derrière vous ne compte le nombre de caractères.
Soyez donc rassurés, le serveur reçoit bien votre mot de passe. Tapez-le comme si de rien n'était.

Si le login et le mot de passe sont bons, vous avez accès à la console du serveur sous Linux comme si vous étiez devant (figure suivante) !

```
sdz@bart: /home/sdz
      ....,I~I=+=+~===+==+=~=+~=7+++..:.
      ....,7================+~+~=====+I:
      ....:I======================+==+?.
      ....=?====================+===+I:.
      ....+?====================+==+?+..
      ....?+=======================I,..
       ...I++=~~==~=====+===+=======~...
      .. .I+===+==+=~~==+===========....
      ...,I========????++?=~=~~=====....
      ...,I=====+=:......~===II+??==....        Bienvenue sur Bart !
      ...:I==~==:........~I:....,~~....
      ...~I===~=..........~...........
      ...:?~====...Z.......=.,......:...
      ..,?7+=~+=:........~I~,ZZ....:...
    ...=~$?~~====+=:...,=?+======.,:.,..
    ...:?===========++++=~======I$?.,...
    ....?+~~=====~~===~~======+?I~7~...
    ...+~~======================I~...
......=+~===~78$~+==~~=~~========I,...
....$$ZZ+~=+=+=~77I====~========+?....
...IIIII7$Z7~==~+==?7I+=+=====+++,....
,~$777777III$7?=??=,....,,,:,......,..
+$7I777I77I77I7Z:...   .. .......
sdz@bart:~$
```



Ne soyez pas surpris par tous les caractères que vous voyez sur ma capture d'écran. On a juste personnalisé le message de bienvenue du serveur « Bart » du Site du Zéro pour qu'il affiche la tête de Bart en lettres à la connexion. ;-) Changer le message de bienvenue se fait facilement mais n'est en général pas très utile.

Dans l'immédiat, vous ne devriez pas avoir à vous connecter à votre ordinateur à distance en utilisant SSH. Tout ce qu'on va faire sera plutôt effectué en local, c'est-à-dire directement sur votre machine. On utilisera la méthode décrite plus haut, à savoir la console en mode graphique.

Au moins, vous savez désormais qu'il est aussi possible de communiquer à distance. Nous reparlerons plus en détails du protocole SSH plus loin dans ce livre, lorsque nous nous intéresserons à l'administration de serveurs.

En résumé

- Bien que rebutante au premier abord, la console nous offre une puissance importante. Elle nous permet d'exécuter des actions simples comme complexes, qui sont parfois impossibles à réaliser avec une interface graphique.
- On peut lancer une console en plein écran avec les touches `Ctrl + Alt + F1` à `F6`, mais il est plus simple aujourd'hui d'ouvrir une console via une interface graphique à l'aide du programme Terminal (sous Unity) ou Konsole (sous KDE).
- On peut se connecter en console à distance à son ordinateur sous Linux grâce au protocole SSH. Cette technique est très fréquemment utilisée pour administrer des serveurs sur Internet équipés de Linux.



Entrer une commande

Chers amis, le grand jour est enfin arrivé ! Vous allez rentrer votre première commande en console !
Ça va, pas trop stressés ?

Je vous rassure, nous allons commencer par des choses très simples pour nous familiariser avec la console. Nous allons vraiment voir le B.A.-BA, le guide de survie élémentaire en quelque sorte.

L'invite de commandes

Je suppose à partir de maintenant que vous avez ouvert une console. Si vous ne savez pas faire, c'est que vous n'avez pas lu le chapitre précédent.
Le mieux, comme je vous l'ai dit, est d'ouvrir une console dans le mode graphique. Le programme **Konsole** sous KDE ou **Terminal** sous Unity fera donc très bien l'affaire (figure suivante).

Terminal de Unity

À partir de maintenant, je vais vous présenter le texte affiché en console dans des encadrés comme celui-ci :

Code : Console

```
mateo21@mateo21-desktop:~$
```

Ça, c'est ce que vous voyez à l'écran. Vous n'avez encore rien écrit, mais l'ordinateur vous dit bonjour à sa manière (bon, O.K., j'avoue que c'est une façon très spéciale de dire bonjour).

Ce que vous voyez là est ce qu'on appelle l'**invite de commandes**. C'est un message qui vous **invite** à rentrer une **commande** en vous donnant par la même occasion une foule d'informations. Cette invite s'affiche avant chaque commande que vous tapez.

Bien : décortiquons cette invite de commandes parce qu'elle est très intéressante.

- `mateo21` : le premier élément est votre pseudonyme. C'est le pseudo sous lequel vous vous êtes loggés. En effet, rappelez-vous : on peut créer plusieurs comptes utilisateurs sous Linux. Il est en général conseillé d'en générer un par

personne susceptible d'utiliser l'ordinateur (un pour chaque membre de la famille, par exemple). Nous verrons plus tard comment rajouter des comptes utilisateurs.
- `@` : ce symbole n'indique rien de particulier. C'est le symbole « at » qui signifie « chez ». Si on lit l'invite de gauche à droite, on doit donc comprendre « mateo21 chez ».
- `mateo21-desktop` : ça, c'est le nom de l'ordinateur sur lequel vous êtes en train de travailler. Dans mon cas il s'appelle `mateo21-desktop`, mais j'aurais pu lui attribuer n'importe quel nom lors de l'installation.
  Par exemple, on a l'habitude de donner le nom d'un membre des Simpson à chacun des serveurs du Site du Zéro : Lisa, Bart, Itchy, Scratchy… Cela permet de savoir de quelle machine on parle quand on dit « Ouh là, Bart est surchargé, il faudrait voir quel est le programme qui ralentit tout ».Si vous suivez toujours, la ligne d'invite de commandes se lit donc « mateo21 chez mateo21-desktop ». En d'autres termes, je suis identifié en tant que `mateo21` sur la machine `mateo21-desktop`.
- **:** : à nouveau, ce symbole ne veut rien dire de spécial, c'est un séparateur.
- **~** : ça, c'est le dossier dans lequel vous vous trouvez actuellement. Vous pouvez naviguer de dossier en dossier dans la console et il est très utile qu'on vous rappelle systématiquement où vous vous trouvez avant chaque commande.
  Pour information, le symbole ~ signifie que vous êtes dans votre dossier personnel, ce qu'on appelle le « home » sous Linux ; c'est l'équivalent du dossier « Mes documents » de Windows. Nous étudierons plus en détail le fonctionnement des dossiers sous Linux dans le prochain chapitre.
- **$** : ce dernier symbole est très important ; il indique votre niveau d'autorisation sur la machine. Il peut prendre deux formes différentes :
  - **$** : signifie que vous êtes en train d'utiliser un compte utilisateur « normal », avec des droits limités (il ne peut pas modifier les fichiers système les plus importants). Mon compte `mateo21` est donc un compte normal avec des droits limités ;
  - **#** : signifie que vous êtes en mode superutilisateur, c'est-à-dire que vous êtes connectés sous le pseudonyme « root ». Le root est l'utilisateur maître qui a le droit de tout faire sur sa machine (même de la détruire !). Nous verrons le mode root plus en détail plus tard ; pour l'instant nous restons dans un compte utilisateur limité, ainsi nous ne risquons pas de faire de bêtise.

Comme vous le voyez, une fois qu'on parle la même langue que l'invite de commandes, on comprend ce qu'elle veut dire !
« Bonjour et bienvenue, vous êtes `mateo21` sur la machine `mateo21-desktop`. Vous vous trouvez actuellement dans votre dossier `home` et possédez des droits utilisateur limités. ~~La température extérieure est de~~. »



Comme un peu tout sous Linux, l'invite de commandes est totalement paramétrable. Vous pouvez la raccourcir si vous trouvez qu'elle est trop longue, ou la rallonger si vous trouvez qu'elle ne donne pas assez d'informations. Vous pouvez en théorie mettre vraiment tout ce que vous voulez dans l'invite, comme par exemple l'heure actuelle (par contre, pour la température extérieure il faudra repasser).
Nous verrons comment changer cela lorsque vous aurez appris à vous servir d'un éditeur de texte !

Commandes et paramètres

On travaille dans la console en tapant ce qu'on appelle des **commandes**. Ces dernières étant nombreuses, vous ne pourrez jamais toutes les connaître… et ce n'est pas le but : le but, c'est que vous sachiez vous servir par cœur de la plupart des commandes « courantes » et, pour les moins courantes, que vous soyez capables d'apprendre à vous en servir en lisant leur **manuel d'utilisation**.

Le manuel d'utilisation est **la véritable bible de tous les linuxiens**. Vous verrez rapidement qu'ils ne jurent que par ça. Pourquoi ? Parce que c'est tout simplement un outil de référence, là où l'on peut trouver la réponse à TOUTES ses questions pour peu qu'on sache lire le manuel et qu'on prenne la peine de le faire. Un chapitre entier vous apprendra à lire le manuel : c'est vraiment très important.



Pour information, il est courant de voir un linuxien dire « RTFM » aux débutants qui posent des questions qu'il estime « simples ». RTFM est l'abréviation de « Read the fucking manual! », ce qui veut dire grosso modo « Lis le p*** de manuel ! ». Ce n'est pas vraiment une insulte en fait, mais plutôt une sorte de… soupir d'exaspération.
Pour éviter qu'on vous dise ça un jour, je vous conseille donc de bien ouvrir vos oreilles quand je vous expliquerai comment utiliser le manuel. ;-)

Une commande simple

Bon : trêve de bavardages, on va rentrer une commande ! Par exemple, tapez `date` puis appuyez sur la touche `Entrée` du clavier.
Le résultat devrait ressembler à cela :

Code : Console

```
mateo21@mateo21-desktop:~$ date
lundi 20 septembre 2010, 15:39:51 (UTC+0200)
```

La première ligne contient l'invite de commandes suivie de la commande que j'ai tapée.
La seconde ligne est la réponse de l'ordinateur à cette commande.

Je suppose que vous avez deviné comme des grands ce que l'on vient de faire : on a demandé quelles étaient la date et l'heure !

Vous en voulez encore ? O.K., alors essayons une toute autre commande : tapez `ls`. C'est l'abréviation de « list », qui signifie « **lister les fichiers et dossiers du répertoire actuel** ».

Code : Console

```
mateo21@mateo21-desktop:~$ ls
Desktop Examples Images
```

Cela signifie que le répertoire actuel est constitué de trois dossiers : `Desktop`, `Examples` et `Images`. En général, le système colore les éléments pour que l'on puisse distinguer facilement les dossiers des fichiers.
Si vous n'avez aucune réponse, c'est que vous êtes dans un dossier qui ne contient ni fichier ni dossier.

Voilà, c'est aussi simple que cela. Une commande est constituée d'un mot et ne contient aucun espace. Dans des cas très simples comme ceux que l'on vient de voir, il suffit juste de taper la commande pour avoir une réponse ; mais dans la quasi-totalité des cas on peut (et parfois on DOIT) rentrer des options, qu'on appelle **paramètres**.

Les paramètres

Les paramètres sont des options que l'on écrit à la suite de la commande. La commande et les paramètres sont séparés par un espace, comme ceci :

Code : Console

```
mateo21@mateo21-desktop:~$ commande parametres
```

Les paramètres peuvent eux-mêmes contenir des espaces, des lettres, des chiffres… un peu de tout, en fait. Il n'y a pas de règle véritable sur la forme des paramètres, mais heureusement les programmeurs ont adopté une sorte de « convention » pour que l'on puisse reconnaître les différents types de paramètres.

Les paramètres courts (une lettre)

Les paramètres les plus courants sont constitués d'une seule lettre précédée d'un tiret. Par exemple :

Code : Console

```
commande -d
```

Si on doit donner plusieurs paramètres, on peut faire comme ceci :

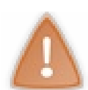
Code : Console

```
commande -d -a -U -h
```

Ou, plus court :

Code : Console

```
commande -daUh
```



Attention à la casse des paramètres (majuscules / minuscules) ! Si vous écrivez `-u`, cela n'a en général pas du tout le même sens que `-U` !

Faisons un essai avec la commande `ls` et rajoutons-lui le paramètre « `a` » (en minuscule) :

Code : Console

```
mateo21@mateo21-desktop:~$ ls -a
.                .gconfd             .mozilla-thunderbird
..               .gimp-2.2           .nautilus
.bash_history    .gksu.lock          .profile
.bash_logout     .gnome              .recently-used
.bashrc          .gnome2             .recently-used.xbel
.config          .gnome2_private     .ssh
Desktop          .gstreamer-0.10     .sudo_as_admin_successful
.dmrc            .gtkrc-1.2-gnome2   .themes
.esd_auth        .ICEauthority       .thumbnails
.evolution       .icons              .Trash
Examples         .lesshst            tutos
.face            .local              .update-manager-core
.fontconfig      .macromedia         .update-notifier
.gaim            .metacity           .Xauthority
.gconf           .mozilla            .xsession-errors
```

Cela affiche tout le contenu du dossier, même les fichiers cachés.
Un « fichier caché » sous Linux est un fichier qui commence par un point. Normalement, si vous vous trouvez dans votre répertoire `home`, vous devriez avoir une bonne flopée de fichiers cachés. Ce sont en général des fichiers de configuration de programmes.

Les paramètres longs (plusieurs lettres)

Les paramètres constitués de plusieurs lettres sont précédés de deux tirets, comme ceci :

Code : Console

```
commande --parametre
```

Cette fois, pas le choix : si vous voulez mettre plusieurs paramètres longs, il faudra ajouter un espace entre chacun d'eux :

Code : Console

```
commande --parametre1 --parametre2
```

On peut aussi combiner les paramètres longs et les paramètres courts dans une commande :

Code : Console

```
commande -daUh --autreparametre
```



Il y a parfois deux écritures possibles pour un paramètre de commande : une version courte et une version longue. Cela permet de vous laisser le choix selon que vous préférez l'une ou l'autre.
Notez que c'est la commande qui décide des paramètres qu'elle accepte : il arrive parfois que certaines ne proposent pas le choix entre une version courte et une longue.

Testons cela sur la commande `ls` avec le paramètre `--all`, qui signifie « tout » en anglais :

Code : Console

```
mateo21@mateo21-desktop:~$ ls --all
.               .gconfd            .mozilla-thunderbird
..              .gimp-2.2          .nautilus
.bash_history   .gksu.lock         .profile
.bash_logout    .gnome             .recently-used
.bashrc         .gnome2            .recently-used.xbel
.config         .gnome2_private    .ssh
Desktop         .gstreamer-0.10    .sudo_as_admin_successful
.dmrc           .gtkrc-1.2-gnome2  .themes
.esd_auth       .ICEauthority      .thumbnails
.evolution      .icons             .Trash
Examples        .lesshst           tutos
.face           .local             .update-manager-core
.fontconfig     .macromedia        .update-notifier
.gaim           .metacity          .Xauthority
.gconf          .mozilla           .xsession-errors
```

Comme vous le voyez, `--all` est un synonyme de `-a`. Ceci illustre ce que je vous disais à l'instant, comme quoi parfois une commande propose deux façons d'utiliser un paramètre : une courte et une longue.

Les valeurs des paramètres

Certains paramètres nécessitent que vous les complétiez avec une valeur. Cela fonctionne différemment selon que vous travaillez avec un paramètre long ou avec un paramètre court.

Avec un paramètre court :

Code : Console

```
commande -p 14
```

… cela indique que l'on associe la valeur `14` au paramètre `p`. Avec ce genre de technique, on peut par exemple faire comprendre à l'ordinateur : « **Je veux voir la liste de tous les fichiers de plus de 14 Mo** ».

Si c'est un paramètre long, on fait en général comme ceci :

Code : Console

```
commande --parametre=14
```

Le résultat sera le même, il est juste plus lisible mais aussi plus long à écrire.

Les autres paramètres

Je vous l'ai dit : il n'y a pas de règle absolue au niveau des paramètres et vous en rencontrerez sûrement qui fonctionnent différemment. Heureusement, les « conventions » que je viens de vous donner sont valables dans la grande majorité des cas, ce qui devrait vous permettre de vous repérer.

Certains paramètres sont donc un peu différents et dépendent vraiment des commandes. Par exemple avec `ls`, si on ajoute le nom d'un dossier (ou sous-dossier), cela affichera le contenu de ce dossier au lieu du contenu du dossier courant :

Code : Console

```
mateo21@mateo21-desktop:~$ ls Examples
Experience ubuntu.ogg   logo-Ubuntu.png          oo-payment-schedule.ods
fables_01_01_aesop.spx  oo-about-these-files.odt oo-presenting-
kubuntu.odp
gimp-ubuntu-splash.xcf  oo-about-ubuntu-ru.rtf   oo-presenting-
ubuntu.odp
kubuntu-leaflet.png     oo-cd-cover.odg          oo-trig.xls
logo-Edubuntu.png       oo-derivatives.doc       oo-welcome.odt
logo-Kubuntu.png        oo-maxwell.odt           ubuntu Sax.ogg
```

Retrouver une commande

Linux propose tellement de commandes différentes qu'il est facile de s'y perdre et d'en oublier une. Personnellement, ça m'arrive très régulièrement, mais ce n'est heureusement pas un drame. En effet, Linux vous propose toute une série de façons de retrouver une commande que vous avez oubliée.

Autocomplétion de commande

Le premier « truc » à connaître, c'est l'autocomplétion de commande. Prenons la commande `date` par exemple : vous êtes un peu tête en l'air et vous ne savez plus comment elle s'écrit. Par contre, vous êtes sûrs des premières lettres de la commande.

Lister les commandes correspondantes

Tapez juste « `da` » dans la console, puis tapez deux fois sur la touche **`Tabulation`** située à gauche de votre clavier. Le résultat sera le suivant :

Code : Console

```
mateo21@mateo21-desktop:~$ da
dash date
mateo21@mateo21-desktop:~$ da
```

En tapant deux fois sur `Tabulation`, vous avez demandé à l'ordinateur la liste des commandes qui commencent par « `da` ». On vous a répondu « `dash` » et « `date` ». Il y a donc deux commandes qui commencent par « `da` », et vous venez de retrouver celle que vous cherchiez, c'est-à-dire « `date` ».

Bien sympathique, l'ordinateur a réécrit l'invite de commandes en dessous ainsi que le début de la commande que vous aviez tapée. Vous n'avez plus qu'à compléter avec les lettres « `te` » qui manquent et à taper `Entrée`, et ce sera bon. :-)

L'autocomplétion

Plus sympa encore, s'il n'y a qu'un seul résultat correspondant à votre recherche, l'ordinateur complètera avec les lettres qui manquent et vous n'aurez plus qu'à taper sur `Entrée` !
Par exemple, il n'y a qu'une commande qui commence par « `dat` ». Tapez donc `dat` dans la console, puis appuyez une seule fois sur **`Tabulation`**. La commande se complète comme par magie.

Trop de commaaaandes !

Parfois, il y a trop de commandes correspondant à votre recherche. Faites un essai un peu brutal : ne rentrez aucun début de commande et faites deux fois `Tab` (`Tabulation`). Cela demande de faire la liste de toutes les commandes disponibles sur votre ordinateur.

Code : Console

```
mateo21@mateo21-desktop:~$
Display all 2173 possibilities? (y or n)
```

Sauvage, n'est-ce pas ?
Il y a 2 173 commandes disponibles sur mon ordinateur. Plus j'installerai de programmes, plus j'aurai de commandes utilisables. N'espérez donc pas toutes les connaître, de nouveaux programmes sortent tous les jours.

À cette question, vous pouvez répondre « `y` » (*yes*) et la liste s'affichera page par page. Quelques raccourcis à connaître quand une liste s'affiche page par page :

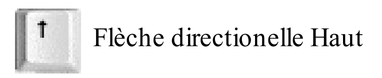
- tapez `Espace` pour passer à la page suivante ;
- tapez `Entrée` pour aller à la ligne suivante ;
- tapez `q` pour arrêter la liste.

Si vous répondez « `n` » (*no*), il ne se passera rien ; c'est dans le cas où vous vous diriez « Ouh là, 2 173 possibilités : autant chercher une aiguille dans une botte de foin… je vais peut-être affiner ma recherche ».

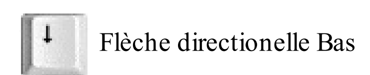
L'historique des commandes

On a très souvent besoin de retrouver une commande que l'on a tapée il y a cinq minutes (ou même cinq secondes). Parfois c'est parce qu'on a oublié la commande, mais c'est souvent aussi parce qu'on a comme moi un énooorme poil dans la main et qu'on a vraiment la flemme de réécrire nous-mêmes la commande en entier.

Ce raccourci vaut de l'or : appuyez sur la flèche directionnelle `Haut` (figure suivante) ; vous verrez apparaître la dernière commande que vous avez tapée.
Si vous appuyez de nouveau sur la flèche directionnelle `Haut`, vous verrez l'avant-dernière commande, puis l'avant-avant-dernière, etc.



Si vous appuyez sur la flèche directionnelle `Bas` (figure suivante), vous reviendrez aux commandes les plus récentes.



C'est ainsi que je peux successivement retrouver les commandes que je viens de taper, dans l'ordre inverse :

- `ls --all` ;
- `ls -a` ;
- `ls` ;
- `date` ;
- etc.

Si vous voulez « remonter » très loin en arrière dans l'historique de vos commandes, pas la peine de taper cent fois sur la flèche

directionnelle `Haut` comme des forcenés.
Il existe la commande `history` qui vous rappelle l'historique des commandes :

Code : Console

```
152  date
153  ls
154  ls -a
155  ls --all
156  history
```

La dernière commande tapée sera toujours `history`, forcément.
Vous remarquerez que les commandes sont numérotées : ainsi, on peut savoir que `date` est la 152ème commande que j'ai tapée dans le terminal.

`Ctrl + R` : rechercher une commande tapée avec quelques lettres

Dans le cas où la flèche directionnelle `Haut` et la commande `history` ne suffiraient pas à retrouver une vieille commande que vous avez tapée, il y a un raccourci super utile : `Ctrl + R`. Appuyez donc sur les touches `Ctrl` et `R` en même temps et l'ordinateur se mettra en mode « recherche d'une commande tapée » (« R » comme **R**echerche).

Là, vous pouvez taper n'importe quelle suite de lettres correspondant à une vieille commande. Par exemple, faites `Ctrl + R` puis tapez « `all` ». Linux retrouve la commande `ls --all` qui contenait justement le mot « `all` ». Vous n'avez plus qu'à taper `Entrée` pour relancer la commande ! :-)

Code : Console

```
(reverse-i-search)`all': ls --all
```

Si ce n'est pas la commande que vous cherchiez, appuyez à nouveau sur `Ctrl + R` pour remonter dans la liste des commandes contenant « `all` ».

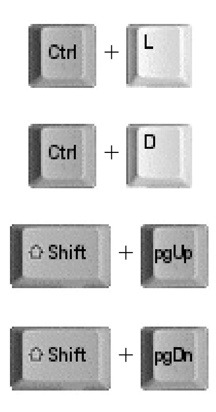
Ça a peut-être l'air bête sur une commande comme ça, mais certaines sont vraiment très longues et c'est un vrai bonheur de ne pas avoir à les réécrire en entier !

Quelques raccourcis clavier pratiques

On ne dirait pas comme ça, mais la console de Linux propose une quantité incroyable de raccourcis clavier. Ce sont des raccourcis qu'on ne peut pas deviner, qu'on a un peu de mal à retenir au début, mais quand on les connaît… waouh ! On devient un peu comme Neo dans *Matrix* en fait, on va très vite.

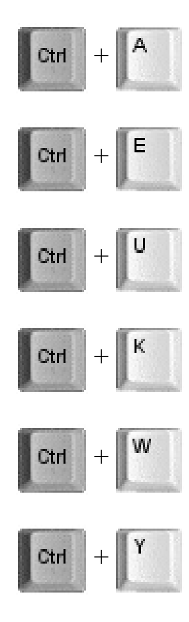
Ces raccourcis ne sont pas intuitifs, mais ça vaut vraiment le coup de les retenir. Les premiers temps vous reviendrez certainement souvent ici pour les consulter (et encore, si vous vous faites un petit pense-bête, ça ne devrait pas arriver), mais petit à petit vous les connaîtrez par cœur !

Commençons par quelques raccourcis généraux qu'il vous faut connaître.



- **`Ctrl + L`** : efface le contenu de la console. Utile pour faire un peu de ménage quand votre console est encombrée, ou quand votre boss passe derrière vous et que vous n'aimeriez pas qu'il voie ce que vous étiez en train de faire.
  À noter qu'il existe aussi une commande, `clear`, qui fait exactement la même chose.
- **`Ctrl + D`** : envoie le message `EOF` (fin de fichier) à la console. Si vous tapez ce raccourci dans une ligne de commande vide (c'est-à-dire sans avoir écrit un début de commande au préalable), cela fermera la console en cours.
  À noter qu'il existe aussi la commande `exit` qui a le même effet.
- **`Shift + PgUp`** : vous permet de « remonter » dans les messages envoyés par la console. En mode graphique, la molette de la souris accomplit aussi très bien cette action.
  La touche `Page Up` est généralement représentée sur votre clavier par une flèche directionnelle `Haut` barrée de plusieurs petites lignes horizontales.
- **`Shift + PgDown`** : pareil, mais pour redescendre.

Les raccourcis suivants sont utiles lorsque vous êtes en train d'écrire une longue commande.



- **`Ctrl + A`** : ramène le curseur au début de la commande. La touche `Origine` a le même effet (elle est située à côté de la touche `Fin` et représentée par une flèche pointant en haut à gauche).
- **`Ctrl + E`** : ramène le curseur à la fin de la ligne de commandes. La touche `Fin` a le même effet.
- **`Ctrl + U`** : supprime tout ce qui se trouve à gauche du curseur. Si celui-ci est situé à la fin de la ligne, cette dernière sera donc supprimée.
- **`Ctrl + K`** : supprime tout ce qui se trouve à droite du curseur. S'il est situé au début de la ligne, celle-ci sera donc totalement supprimée.
- **`Ctrl + W`** : supprime le premier mot situé à gauche du curseur. Un « mot » est séparé par des espaces ; on s'en sert en général pour supprimer le paramètre situé à gauche du curseur.
- **`Ctrl + Y`** : si vous avez supprimé du texte avec une des commandes `Ctrl + U`, `Ctrl + K` ou `Ctrl + W` qu'on vient de voir, alors le raccourci `Ctrl + Y` « collera » le texte que vous venez de supprimer. C'est donc un peu comme un couper-coller.

Nous allons arrêter là la liste pour le moment. Ça vous fait déjà pas mal de choses à retenir.
Il existe en fait beaucoup d'autres raccourcis clavier, mais je vous les montrerai au fur et à mesure, quand nous en aurons besoin.

Même si c'est un peu fastidieux, je vous conseille de vous entraîner à les retenir par cœur. Vous serez vraiment beaucoup plus efficaces lorsque vous les connaîtrez !

En résumé

- La console affiche une invite de commandes au début de la ligne. Cette invite rappelle votre nom d'utilisateur, le nom de la machine ainsi que le dossier dans lequel vous vous trouvez.
- On rentre des **commandes** dans la console pour demander à l'ordinateur d'exécuter des actions.
- Chaque commande peut être complétée de **paramètres** qui agissent comme des options pour modifier l'action de la commande.
- Les paramètres sont généralement constitués d'une lettre précédée d'un tiret (`-a`) ou de plusieurs lettres précédées de deux tirets (`--all`).
- Après avoir saisi les premières lettres d'une commande, on peut compléter son nom à l'aide de la touche `Tabulation`.
- On peut retrouver les commandes précédentes à l'aide des flèches directionnelles `Haut` et `Bas` ou encore effectuer une

recherche parmi les commandes précédentes avec `Ctrl + R`.

- Il existe de nombreux autres raccourcis clavier qu'il est recommandé de connaître pour pouvoir profiter pleinement de la console.

 La structure des dossiers et fichiers

Ahhh, les fichiers sous Linux, tout un programme.
Vous croyez savoir ce que sont les fichiers et dossiers ? Vous croyez que votre disque dur s'appelle `C:` ? Que le lecteur CD c'est `D:` ou peut-être `E:` ?

Les choses ne fonctionnent pas du tout de la même manière sous Linux et sous Windows. Or, savoir comment se déplacer de dossier en dossier et savoir lister les fichiers, c'est quand même sacrément important ! C'est pour cela que nous allons voir ensemble le fonctionnement des fichiers sous Linux dès maintenant.

Organisation des dossiers

Le système qui gère les fichiers sous Linux est un peu déroutant au début, surtout quand on est habitué à celui de Windows. En effet, ici vous ne trouverez pas de `C:\`, `D:\` ou que sais-je encore. Les fichiers sont organisés d'une manière complètement différente.

Au lieu de séparer chaque disque dur, lecteur CD, lecteur de disquettes, lecteur de carte mémoire… Linux place en gros tout au même endroit.



Mais comment on fait pour savoir si le dossier dans lequel on est appartient au premier disque dur, au second disque dur, au lecteur CD… ? C'est le bazar, non ?

C'est ce qu'on pourrait croire au premier abord, mais en fait c'est juste une autre façon de penser la chose. ;-)

Deux types de fichiers

Pour faire simple, il existe deux grands types de fichiers sous Linux :

- **les fichiers classiques** : ce sont les fichiers que vous connaissez, ça comprend les fichiers texte (`.txt`, `.doc`, `.odt`…), les sons (`.wav`, `.mp3`, `.ogg`), mais aussi les programmes. Bref, tout ça, ce sont des fichiers que vous connaissez et que vous retrouvez dans Windows ;
- **les fichiers spéciaux** : certains autres fichiers sont spéciaux car ils **représentent** quelque chose. Par exemple, votre lecteur CD est un fichier pour Linux. Là où Windows fait la distinction entre ce qui est un fichier et ce qui ne l'est pas, Linux, lui, dit que **tout est un fichier**. C'est une conception très différente, un peu déroutante comme je vous l'ai dit, mais pas de panique, vous allez vous y faire.

La racine

Dans un système de fichiers, il y a toujours ce qu'on appelle une racine, c'est-à-dire un « **gros dossier de base qui contient tous les autres dossiers et fichiers** ».

Sous Windows, il y a en fait plusieurs racines. `C:\` est la racine de votre disque dur, `D:\` est la racine de votre lecteur CD (par exemple).
Sous Linux, **il n'y a qu'une et une seule racine** : « `/` ». Comme vous le voyez, il n'y a pas de lettre de lecteur car justement, Linux ne donne pas de nom aux lecteurs comme le fait Windows. Il dit juste « **La base, c'est /** ».



Il n'y a pas de dossier de plus haut niveau que `/`, c'est-à-dire qu'il n'existe pas de dossier qui contienne le dossier `/`. Quand on est à la racine, on ne peut pas remonter en arrière car… on est déjà tout au début.

Architecture des dossiers

Sous Windows, un dossier peut être représenté de la manière suivante : `C:\Program Files\Winzip`. On dit que `Winzip` est un sous-dossier du dossier `Program Files`, lui-même situé à la racine.
Vous noterez que c'est l'antislash \ (aussi appelé *backslash*) qui sert de séparateur aux noms de dossiers.

Sous Linux, c'est au contraire le / qui sert de séparateur.
Comme je vous l'ai dit, il n'y a pas de `C:` sous Linux, la racine (le début) s'appelant juste `/`.

Le dossier de notre superprogramme ressemblerait plutôt à quelque chose comme cela : `/usr/bin/`. On dit que `bin` est un

sous-dossier du dossier `usr`, lui-même situé à la racine.



Linux gère sans problème les noms de fichiers et dossiers contenant des espaces, des accents et des majuscules. Toutefois, vous remarquerez que la plupart du temps on préfère les éviter. On trouve ainsi plutôt des noms tout en minuscules sans accents ni espaces, comme `usr`, `bin`, `apache`, etc.
Souvenez-vous qu'il n'est pas obligatoire de nommer vos fichiers en suivant la même règle, mais la plupart des programmes que vous installerez préfèrent utiliser des noms tout en minuscules sans espaces ni accents, ne soyez donc pas surpris.

Les dossiers de la racine

Sous Windows, on a l'habitude de trouver souvent les mêmes dossiers à la racine : `Documents and Settings`, `Program Files`, `Windows`...
Sous Linux, vous vous en doutez, les dossiers sont complètement différents. Et l'on ne risque pas de trouver de dossier qui s'appelle Windows !

Je vais vous faire ici la liste des dossiers les plus courants que l'on retrouve à chaque fois à la racine de Linux. La description de chaque dossier sera rapide, mais c'est juste pour que vous puissiez vous repérer au début. ;-)



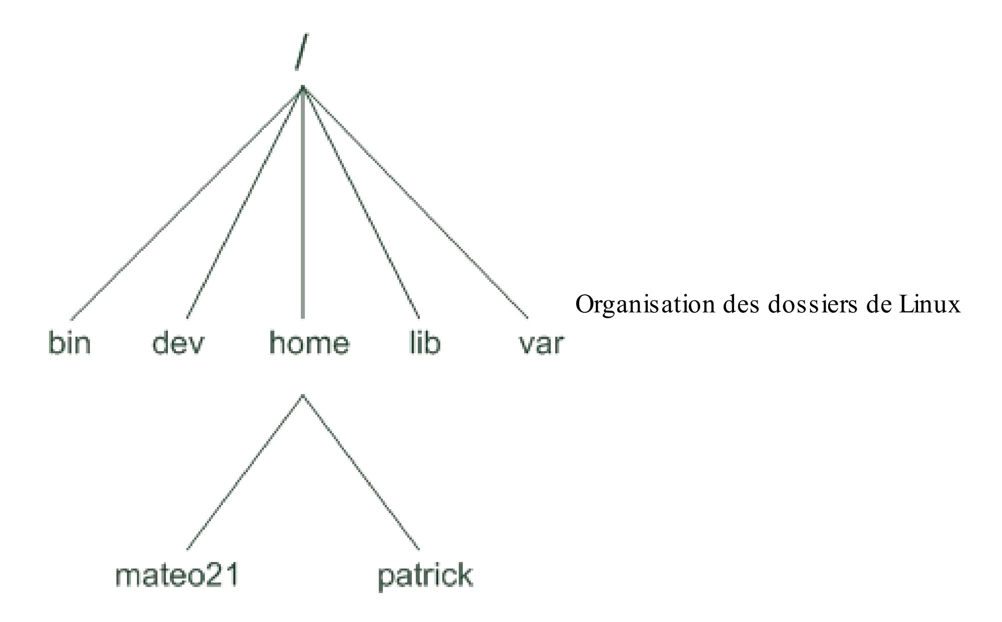
Il n'est PAS nécessaire de retenir cette liste par cœur. D'ailleurs je n'ai mis que les dossiers principaux, et elle est quand même longue.
Servez-vous-en juste si vous avez besoin de savoir à quoi correspond grosso-modo tel ou tel dossier, mais ne vous en faites pas si vous ne maîtrisez pas à fond le sens de chacun de ces dossiers.

- **bin** : contient des programmes (exécutables) susceptibles d'être utilisés par tous les utilisateurs de la machine.
- **boot** : fichiers permettant le démarrage de Linux.
- **dev** : fichiers contenant les périphériques. En fait – on en reparlera plus tard – ce dossier contient des sous-dossiers qui « représentent » chacun un périphérique. On y retrouve ainsi par exemple le fichier qui représente le lecteur CD.
- **etc** : fichiers de configuration.
- **home** : répertoires personnels des utilisateurs. On en a déjà parlé un peu avant : c'est dans ce dossier que vous placerez vos fichiers personnels, à la manière du dossier `Mes documents` de Windows.
  Chaque utilisateur de l'ordinateur possède son dossier personnel. Par exemple, dans mon cas mon dossier personnel se trouve dans `/home/mateo21/`. S'il y avait un autre utilisateur (appelons-le Patrick) sur mon ordinateur, il aurait eu droit lui aussi à son propre dossier : `/home/patrick/`.
- **lib** : dossier contenant les bibliothèques partagées (généralement des fichiers `.so`) utilisées par les programmes. C'est en fait là qu'on trouve l'équivalent des `.dll` de Windows.
- **media** : lorsqu'un périphérique amovible (comme une carte mémoire SD ou une clé USB) est inséré dans votre ordinateur, Linux vous permet d'y accéder à partir d'un sous-dossier de `media`. On parle de **montage**.
- **mnt** : c'est un peu pareil que `media`, mais pour un usage plus temporaire.
- **opt** : répertoire utilisé pour les *add-ons* de programmes.
- **proc** : contient des informations système.
- **root** : c'est le dossier personnel de l'utilisateur « root ». Normalement, les dossiers personnels sont placés dans `home`, mais celui de « root » fait exception. En effet, comme je vous l'ai dit dans le chapitre précédent, « root » est le superutilisateur, le « chef » de la machine en quelque sorte. Il a droit à un espace spécial.
- **sbin** : contient des programmes système importants.
- **tmp** : dossier temporaire utilisé par les programmes pour stocker des fichiers.
- **usr** : c'est un des plus gros dossiers, dans lequel vont s'installer la plupart des programmes demandés par l'utilisateur.
- **var** : ce dossier contient des données « variables », souvent des *logs* (traces écrites de ce qui s'est passé récemment sur l'ordinateur).

Cette liste de dossiers est en fait présente sur tous les OS de type Unix, et pas seulement sous Linux.
Encore une fois, ne retenez pas tout ça. C'est juste pour vous donner une idée de ce que contiennent les dossiers à la racine de Linux, car je sais que c'est une question qu'on se pose souvent quand on débute.

Schéma résumé de l'architecture

Pour que vous vous y repériez correctement, sachez qu'on peut présenter l'organisation des dossiers de Linux comme le suggère la figure suivante.



Organisation des dossiers de Linux

La racine tout en haut est / ; elle contient plusieurs dossiers, qui contiennent chacun eux-mêmes plusieurs dossiers, qui contiennent des dossiers et fichiers, etc.

pwd & which : où… où suis-je ?

Le nombre de dossiers et de fichiers présents après l'installation d'Ubuntu est tellement grand qu'il serait facile de s'y perdre. Un grand nombre de programmes sont en effet préinstallés pour que vous puissiez profiter rapidement des possibilités de Linux. Ne comptez donc pas sur moi pour vous faire la liste complète des dossiers et fichiers que vous possédez, ce n'est pas réaliste.

En revanche, je vais vous apprendre maintenant à vous repérer dans l'arborescence des dossiers. Vous saurez alors à tout moment où vous êtes sur votre disque. C'est un peu comme avoir une carte routière, en quelque sorte !

pwd : afficher le dossier actuel

Lorsque vous ouvrez la console pour la première fois, Linux vous place dans votre dossier personnel, votre `home`. En l'occurrence dans mon cas, le dossier dans lequel je serai placé sera `/home/mateo21`.

Normalement, l'invite de commandes vous indique le nom du dossier dans lequel vous vous trouvez :

Code : Console

```
mateo21@mateo21-desktop:~$
```

Si vous vous souvenez bien, le nom du dossier est situé entre le « : » et le « $ ». Donc ici, on se trouve dans le dossier « ~ ».



Rappel : je l'ai dit dans le chapitre précédent mais ça ne fait pas de mal de le répéter, sous Linux le symbole ~ est un synonyme de votre dossier personnel. Chez moi cela signifie donc `/home/mateo21`.

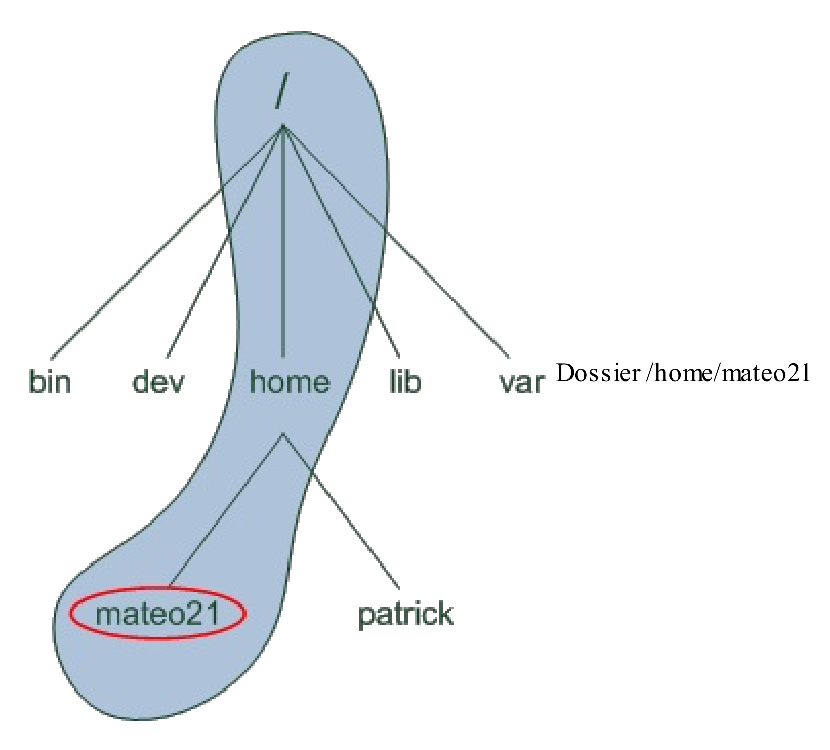
Cette indication de l'invite de commandes est pratique mais il faut savoir qu'il y a un autre moyen de connaître le nom du dossier actuel. C'est la commande **pwd**.
`pwd` est l'abréviation de « Print Working Directory », c'est-à-dire « Afficher le dossier actuel ».

C'est une commande très simple qui ne prend aucun paramètre (on commence doucement, hein !), vous pouvez la tester :

Code : Console

```
mateo21@mateo21-desktop:~$ pwd
/home/mateo21
```

Cela confirme bien ce que je vous disais : je me trouve en ce moment dans le dossier `/home/mateo21` (figure suivante).



Dossier /home/mateo21

À tout moment, si vous vous sentez perdus dans les méandres des dossiers, souvenez-vous de la commande `pwd`, elle vous dira où vous êtes ! ;-)

`which` : connaître l'emplacement d'une commande

Même si cette commande ne nous est pas indispensable, j'ai pensé que c'était une bonne idée de vous la montrer dès le début afin que vous puissiez vous familiariser un peu plus encore avec le système de fichiers de Linux.

Alors, que fait cette commande ? Elle vous permet de localiser la position du programme correspondant à une commande.
Je m'explique : chaque commande sous Linux correspond à un programme. Ainsi, `pwd` qu'on vient de voir **est** un programme.

Une commande n'est rien d'autre qu'un programme qu'on peut appeler n'importe quand et n'importe où dans la console.

La commande `which` prend un paramètre : le nom de la commande dont vous voulez connaître l'emplacement.
Testons sur `pwd` :

Code : Console

```
mateo21@mateo21-desktop:~$ which pwd
/bin/pwd
```

`pwd` se trouve donc dans le dossier `/bin/` ! Le « pwd » à la fin n'est pas un dossier mais le nom du programme lui-même.



Vous noterez que les programmes sous Linux ne possèdent en général pas d'extension (contrairement à Windows où l'extension utilisée est en général `.exe`).

Tous les programmes ne sont pas situés dans un même dossier. Pour vous en rendre compte, testez l'emplacement d'une autre commande… tenez, par exemple la commande `which` !

On va donc devoir écrire `which which` dans la console (oui, je sais, je suis un gros tordu.) :

Code : Console

```
mateo21@mateo21-desktop:~$ which which
/usr/bin/which
```

Cette fois, le programme ne se trouve pas dans `/bin/` mais dans `/usr/bin/` !

ls : lister les fichiers et dossiers

`ls` est une des toutes premières commandes que nous avons essayées dans le chapitre précédent. Nous allons rentrer ici plus dans le détail de son fonctionnement (et de ses nombreux paramètres…).

Commençons par taper « ls » sans paramètre depuis notre dossier personnel (oui : j'ai créé quelques dossiers pour mon usage personnel, ne vous étonnez pas si vous n'avez pas les mêmes.) :

Code : Console

```
mateo21@mateo21-desktop:~$ ls
Desktop  Examples  images  log  tutos
```

Ubuntu active la coloration des fichiers et dossiers par défaut, vous devriez donc voir des couleurs chez vous. Les dossiers apparaissent en bleu foncé. Vous remarquerez que le dossier `Examples` est en bleu clair : cela signifie que c'est un raccourci vers un dossier qui se trouve en fait ailleurs sur le disque.



Si la couleur ne s'affiche pas, vous pouvez rajouter le paramètre `--color=auto`, comme ceci : `ls --color=auto`.
Si vous ne voulez pas de la couleur au contraire, essayez le paramètre `--color=none`.
Pour éviter d'avoir à taper à chaque fois ce long paramètre, il faut modifier un fichier de configuration, mais on verra cela plus tard.

La commande `ls` accepte un grand nombre de paramètres. Ça ne sert à rien que je vous fasse la liste complète ici (ce serait bien trop long) ; par contre, je vais vous faire découvrir les paramètres les plus utiles. Ça vous permettra de vous entraîner à utiliser et combiner des paramètres !

-a : afficher tous les fichiers et dossiers cachés

Sous Linux, on peut « cacher » des fichiers et dossiers. Ce n'est pas une protection, car on peut toujours les réafficher si on veut, mais ça évite d'encombrer l'affichage de la commande `ls`.

Votre dossier `home` est un très bon exemple car il est rempli de fichiers et dossiers cachés. En ajoutant le paramètre `-a`, on peut voir tous ces fichiers et dossiers cachés :

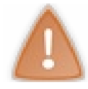
Code : Console

```
mateo21@mateo21-desktop:~$ ls -a
.                                .gnome                .nano_history
..                               .gnome2               .nautilus
.armagetron                      .gnome2_private       .openoffice.org2
.bash_history                    .gnome_private        .pgadmin3
.bash_logout                     .gstreamer-0.10       .pgpass
.bashrc                          .gtkrc-1.2-gnome2     .profile
.blender                         .gweled               .qt
.config                          .ICEauthority         .recently-used
.DCOPserver_mateo21-desktop__0   .icons                .recently-used.xbel
.DCOPserver_mateo21-desktop_:0   images                .ssh
Desktop                          .inkscape             .sudo_as_admin_success
.dmrc                            .java                 .themes
.emilia                          .jedit                .thumbnails
.esd_auth                        .kde                  .Trash
.evolution                       .lesshst              .tsclient
```

```
Examples                         .lgames                  tutos
.face                            .local                   .update-manager-core
.fontconfig                      log                      .update-notifier
.gaim                            .macromedia              .vlc
.gconf                           .mcop                    .wormux
.gconfd                          .mcoprc                  .Xauthority
.geany                           .metacity                .xine
.gimp-2.2                        .mozilla                 .xsession-errors
.gksu.lock                       .mozilla-thunderbird
```

Vous comprenez peut-être mieux maintenant pourquoi tous ces fichiers et dossiers sont cachés : c'est encombrant.
Certains éléments commençant par un point « . » sont des dossiers, d'autres sont des fichiers. La meilleure façon de faire la distinction est de comparer les couleurs : les dossiers en bleu, le reste dans la couleur par défaut (par exemple, le blanc ou le noir).

Les deux premiers éléments sont assez intrigants : « . » et « .. ». Le premier représente en fait le dossier actuel, et « .. » représente le dossier parent, c'est-à-dire le dossier précédent dans l'arborescence. Par exemple, là je suis dans `/home/mateo21`, « .. » représente donc le dossier `/home`.



Le paramètre `-A` (un « A » majuscule au lieu d'un « a » minuscule) a pratiquement la même signification : cela affiche la même chose sauf ces éléments « . » et « .. ». Comme quoi il faut faire attention aux majuscules !

-F : indique le type d'élément

Ce paramètre est surtout utile pour ceux qui n'ont pas affiché la couleur dans la console (ou n'en veulent pas).
Il rajoute à la fin des éléments un symbole pour qu'on puisse faire la distinction entre les dossiers, fichiers, raccourcis…

Code : Console

```
mateo21@mateo21-desktop:~$ ls -F
Desktop/  Examples@  images/  log/  tutos/
```

Grâce à ça on peut voir que tous les éléments sont des dossiers, sauf `Examples` qui est un raccourci (d'où la présence du `@`).

-l : liste détaillée

Le paramètre `-l` (la lettre « L » en minuscule) est un des plus utiles. Il affiche une liste détaillant chaque élément du dossier :

Code : Console

```
mateo21@mateo21-desktop:~$ ls -l
total 16
drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-24 17:22 Desktop
lrwxrwxrwx 1 mateo21 mateo21   26 2007-09-19 18:31 Examples -
> /usr/share/example-content
drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-25 15:17 images
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-25 11:11 log
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-19 19:51 tutos
```

Il y a un élément par ligne.
Chaque colonne a sa propre signification. De gauche à droite :

1. droits sur le fichier (on fera un chapitre entier pour expliquer comment fonctionnent les droits sous Linux) ;
2. nombre de liens physiques (cela ne nous intéresse pas ici) ;

3. nom de la personne propriétaire du fichier (là, c'est moi !). Si le fichier avait été créé par quelqu'un d'autre, par exemple Patrick, on aurait vu son nom à la place ;
4. groupe auquel appartient le fichier (on en reparlera dans le chapitre sur les droits). Il se peut que le nom du groupe soit le même que celui du propriétaire ;
5. taille du fichier, en octets ;
6. date de dernière modification ;
7. nom du fichier (ou dossier).



Vous noterez aussi que dans le cas du raccourci (on parle de **lien symbolique**), la commande nous précise vers où pointe le raccourci (en l'occurrence `/usr/share/example-content`).

`-h` : afficher la taille en Ko, Mo, Go…

Quand on fait un `ls -l`, la taille est affichée en octets. Seulement, ce n'est parfois pas très lisible. Par exemple :

Code : Console

```
mateo21@mateo21-desktop:~/Examples$ ls -l
total 9500
-rw-r--r-- 1 root root 3576296 2007-04-03 17:05 Experience ubuntu.ogg
-rw-r--r-- 1 root root  229674 2007-04-03 17:05 fables_01_01_aesop.spx
-rw-r--r-- 1 root root  848013 2007-04-03 17:05 gimp-ubuntu-splash.xcf
-rw-r--r-- 1 root root 1186219 2007-04-03 17:05 kubuntu-leaflet.png
-rw-r--r-- 1 root root   47584 2007-04-03 17:05 logo-Edubuntu.png
```

Si vous rajoutez le paramètre `h` (« h » pour *Human Readable*, c'est-à-dire « lisible par un humain »), vous obtenez des tailles de fichiers beaucoup plus lisibles (normal, vous êtes des humains) :

Code : Console

```
mateo21@mateo21-desktop:~/Examples$ ls -lh
total 9,3M
-rw-r--r-- 1 root root 3,5M 2007-04-03 17:05 Experience ubuntu.ogg
-rw-r--r-- 1 root root 225K 2007-04-03 17:05 fables_01_01_aesop.spx
-rw-r--r-- 1 root root 829K 2007-04-03 17:05 gimp-ubuntu-splash.xcf
-rw-r--r-- 1 root root 1,2M 2007-04-03 17:05 kubuntu-leaflet.png
-rw-r--r-- 1 root root  47K 2007-04-03 17:05 logo-Edubuntu.png
```

Grâce à ça, on voit alors bien que le fichier `Experience ubuntu.ogg` fait 3,5 Mo, `logo-Edubuntu.png` fait 47 Ko, etc.

`-t` : trier par date de dernière modification

Voilà une option dont l'intérêt est sous-estimé ! `-t` permet en effet de trier par date de dernière modification, au lieu de trier par ordre alphabétique comme cela est fait par défaut.
On voit ainsi en premier le dernier fichier que l'on a modifié, et en dernier celui auquel on n'a pas touché depuis le plus longtemps :

Code : Console

```
mateo21@mateo21-desktop:~$ ls -lt
total 16
drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-25 15:17 images
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-25 11:11 log
drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-24 17:22 Desktop
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-19 19:51 tutos
lrwxrwxrwx 1 mateo21 mateo21   26 2007-09-19 18:31 Examples -
```

```
> /usr/share/example-content
```

De toute évidence, le dernier fichier (ici, c'est un dossier) modifié est « images ». En revanche, comme je n'ai jamais touché à « Examples », il est normal qu'il apparaisse en dernier.

En pratique, je combine `-t` avec `-r` qui renverse l'ordre d'affichage des fichiers. Je préfère en effet avoir le dernier fichier modifié en bas de la liste, c'est plus pratique à l'usage dans la console.
Et comme je suis un gros bourrin, je combine un peu tous les paramètres que l'on vient de voir, ce qui donne un beau `ls -larth` qui contient toutes les options que j'aime. ;-)

Code : Console

```
mateo21@mateo21-desktop:~$ ls -larth
total 380K
-rw-------  1 mateo21 mateo21   26 2007-09-19 16:40 .dmrc
-rw-r--r--  1 mateo21 mateo21   89 2007-09-19 16:40 .gtkrc-1.2-gnome2
-rw-------  1 mateo21 mateo21   16 2007-09-19 16:40 .esd_auth
drwx------  2 mateo21 mateo21 4,0K 2007-09-19 16:40 .update-notifier
lrwxrwxrwx  1 mateo21 mateo21   26 2007-09-19 18:31 Examples -
> /usr/share/example-content
-rw-r--r--  1 mateo21 mateo21  220 2007-09-19 18:31 .bash_logout
drwxr-xr-x  4 root    root    4,0K 2007-09-19 18:31 ..
drwxr-xr-x 10 mateo21 mateo21 4,0K 2007-09-25 16:03 .jedit
-rw-r--r--  1 mateo21 mateo21 1,1K 2007-09-25 16:03 .pgadmin3
drwxr-xr-x 47 mateo21 mateo21 4,0K 2007-09-25 16:03 .
-rw-------  1 mateo21 mateo21 1,8K 2007-09-25 16:38 .bash_history
-rw-------  1 mateo21 mateo21  17K 2007-09-25 16:52 .recently-used
drwx------  2 mateo21 mateo21 4,0K 2007-09-25 16:54 .gconfd
-rw-------  1 mateo21 mateo21   39 2007-09-25 17:18 .lesshst
-rw-r--r--  1 mateo21 mateo21  53K 2007-09-25 17:21 .xsession-errors
```



Note : j'ai volontairement réduit cette liste car il y a beaucoup de fichiers dans mon `home`. En pratique la liste est beaucoup plus grande.

Le fichier caché « .xsession-errors » est donc le dernier qui a été modifié dans ce dossier sur mon ordinateur.



Plutôt que d'avoir à réécrire `ls -larth` à chaque fois (c'est un peu long), on peut créer un alias, c'est-à-dire une commande synonyme. Par exemple, j'ai créé l'alias `ll` (deux fois « L ») qui est automatiquement transformé par Linux en `ls -larth`.
On verra comment créer des alias lorsqu'on saura se servir d'un éditeur de fichiers.

cd: changer de dossier

Bon : mine de rien, depuis tout à l'heure on est coincé dans notre dossier `home` et on aimerait bien bouger de là.
Le moment est venu de déplacer le navire, moussaillon !

La commande que nous allons étudier ici s'appelle `cd`, abréviation de *Change Directory* (changer de dossier). C'est une commande très importante que vous allez utiliser quelques milliers de fois dans votre vie (au moins).

Contrairement à `ls`, la commande `cd` ne prend pas plein de paramètres mais juste un seul : le nom du dossier dans lequel vous souhaitez aller.

Si on veut aller à la racine, il suffit de taper `cd /` :

Code : Console

```
mateo21@mateo21-desktop:~$ cd /
mateo21@mateo21-desktop:/$ pwd
/
```

Après avoir tapé `cd /`, on se retrouve à la racine. L'invite de commandes a changé et le ~ a été remplacé par un `/`. Si vous êtes sceptiques, un petit coup de `pwd` devrait vous confirmer que vous êtes bien dans `/`.

Bien ! Listons les fichiers et dossiers contenus dans `/` :

Code : Console

```
mateo21@mateo21-desktop:/$ ls -F
bin/    dev/   initrd/           lib/          mnt/   root/  sys/  var/
boot/   etc/   initrd.img@       lost+found/   opt/   sbin/  tmp/  vmlinuz@
cdrom@  home/  initrd.img.old@   media/        proc/  srv/   usr/  vmlinuz.old@
```

Vous y retrouvez un grand nombre de dossiers que je vous ai décrits au début du chapitre.
Allons dans le sous-dossier `usr` :

Code : Console

```
mateo21@mateo21-desktop:/$ cd usr
```

Voyons voir ce qu'il y a là-dedans…

Code : Console

```
mateo21@mateo21-desktop:/usr$ ls -F
bin/  games/  include/  lib/  local/  sbin/  share/  src/  X11R6/
```

Chez moi, il n'y a que des dossiers. Hummm, le dossier `games` m'intrigue, voyons voir ce que j'ai comme jeux :

Code : Console

```
mateo21@mateo21-desktop:/usr$ cd games
mateo21@mateo21-desktop:/usr/games$
```

Schématiquement, on vient de faire ce qui est illustré dans la figure suivante.

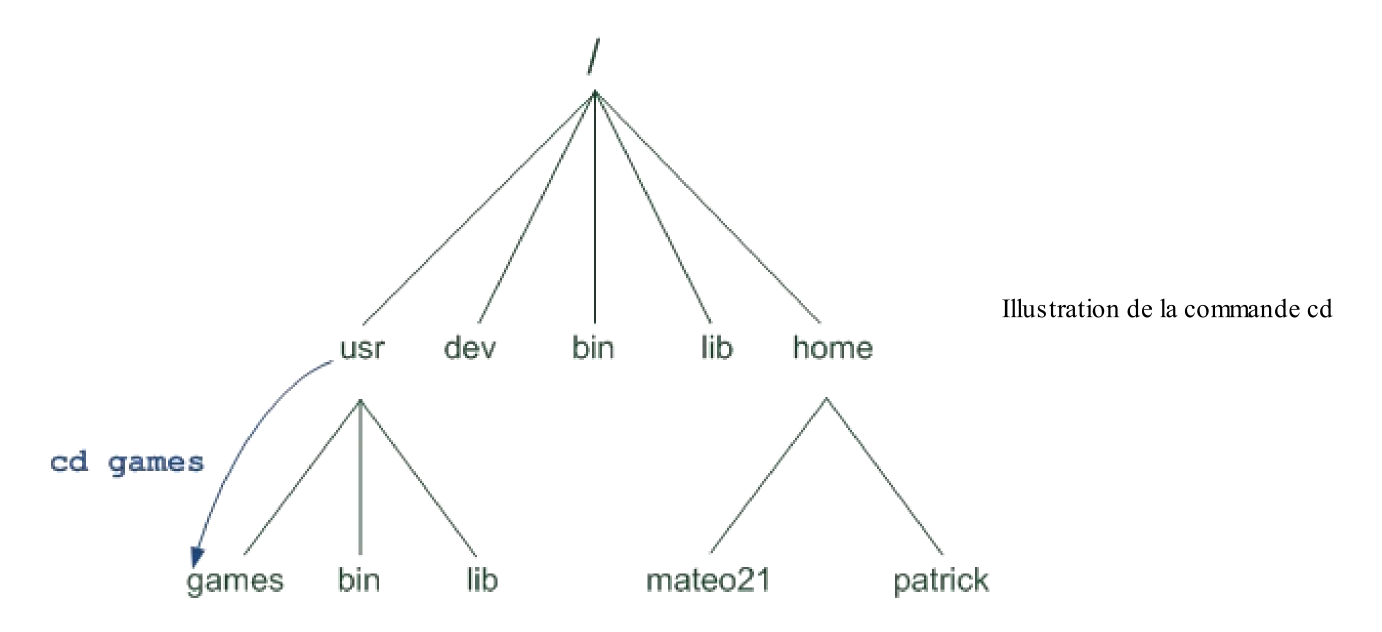


Illustration de la commande cd

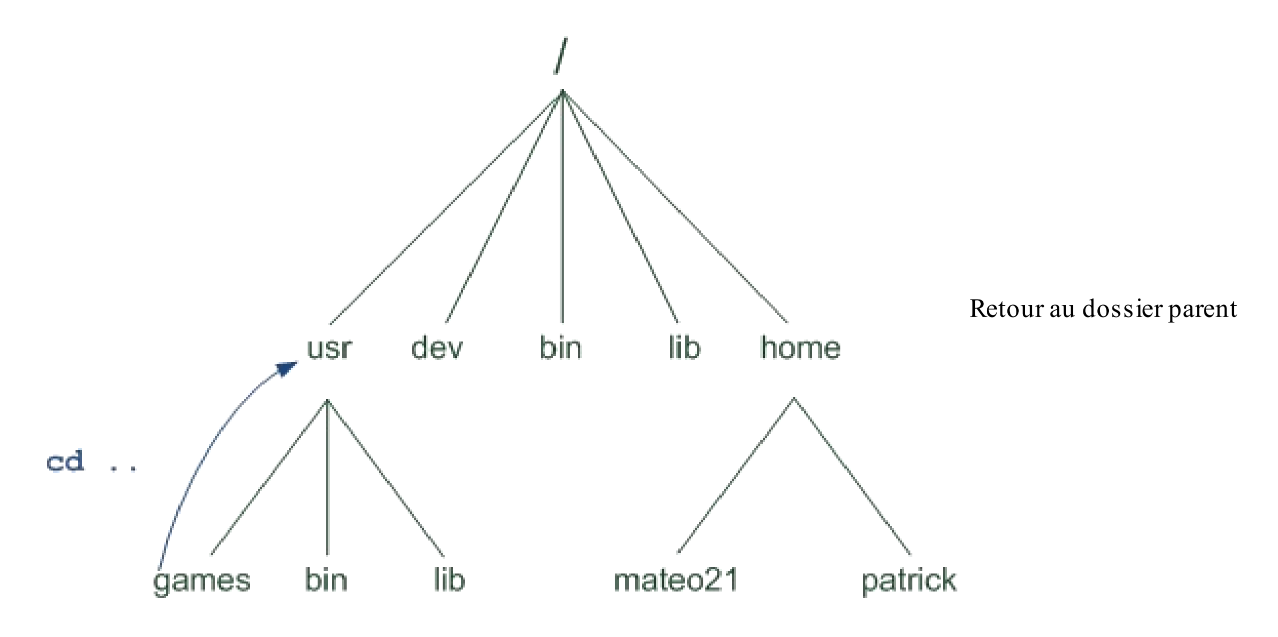
Supposons maintenant que j'aie envie de revenir au dossier précédent, aussi appelé dossier parent, c'est-à-dire `/usr`. Comment je fais ?

Il faut utiliser les deux points comme ceci :

Code : Console

```
mateo21@mateo21-desktop:/usr/games$ cd ..
mateo21@mateo21-desktop:/usr$
```

Et hop là, on est revenu au dossier parent ! (figure suivante.)



Retour au dossier parent

Si on avait voulu reculer de deux dossiers parents, on aurait écrit `../..` (« reviens en arrière, puis reviens en arrière »). Cela nous aurait ramené à la racine :

Code : Console

```
mateo21@mateo21-desktop:/usr/games$ cd ../..
mateo21@mateo21-desktop:/$
```



Eh mais en fait, il y a plusieurs façons d'aller dans un dossier, non ? Tout à l'heure, on est allé à la racine en tapant `cd /`, et maintenant en tapant `cd ../..`
C'est quoi cette affaire ?

Il y a en fait deux façons de changer de dossier : en indiquant un **chemin relatif**, ou en indiquant un **chemin absolu**.

Les chemins relatifs

Un chemin relatif est un chemin qui dépend du dossier dans lequel vous vous trouvez. Tout à l'heure, on est allé dans le sous-dossier `games` de `/usr` en tapant juste son nom :

Code : Console

```
mateo21@mateo21-desktop:/usr$ cd games
```

En faisant cela, on utilise un chemin relatif, c'est-à-dire relatif au dossier actuel. Quand on met juste le nom d'un dossier comme ici, cela indique que l'on veut aller dans un sous-dossier.

Si on fait `cd games` depuis la racine, ça va planter :

Code : Console

```
mateo21@mateo21-desktop:/$ cd games
bash: cd: games: Aucun fichier ou répertoire de ce type
```

Je crois que le message d'erreur est assez clair : il n'y a aucun dossier `games` dans `/`.

Pour se rendre dans `games`, il faut d'abord indiquer le dossier qui le contient (`usr`) :

Code : Console

```
mateo21@mateo21-desktop:/$ cd usr/games
mateo21@mateo21-desktop:/usr/games$
```

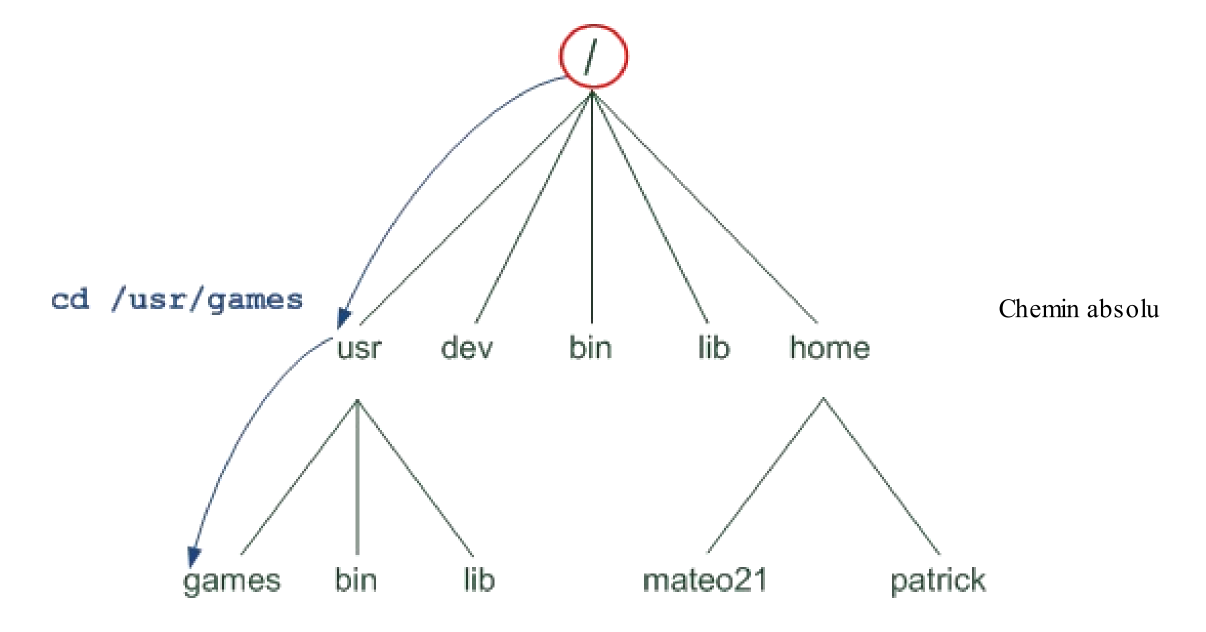
Les chemins absolus

Contrairement aux chemins relatifs, les chemins absolus fonctionnent quel que soit le dossier dans lequel on se trouve.

Un chemin absolu est facile à reconnaître : il commence toujours par la racine (`/`). Vous devez ensuite faire la liste des dossiers dans lesquels vous voulez entrer. Par exemple, supposons que je sois dans `/home/mateo21` et que je souhaite aller dans `/usr/games`. Avec un chemin absolu :

Code : Console

```
mateo21@mateo21-desktop:~$ cd /usr/games
mateo21@mateo21-desktop:/usr/games$
```



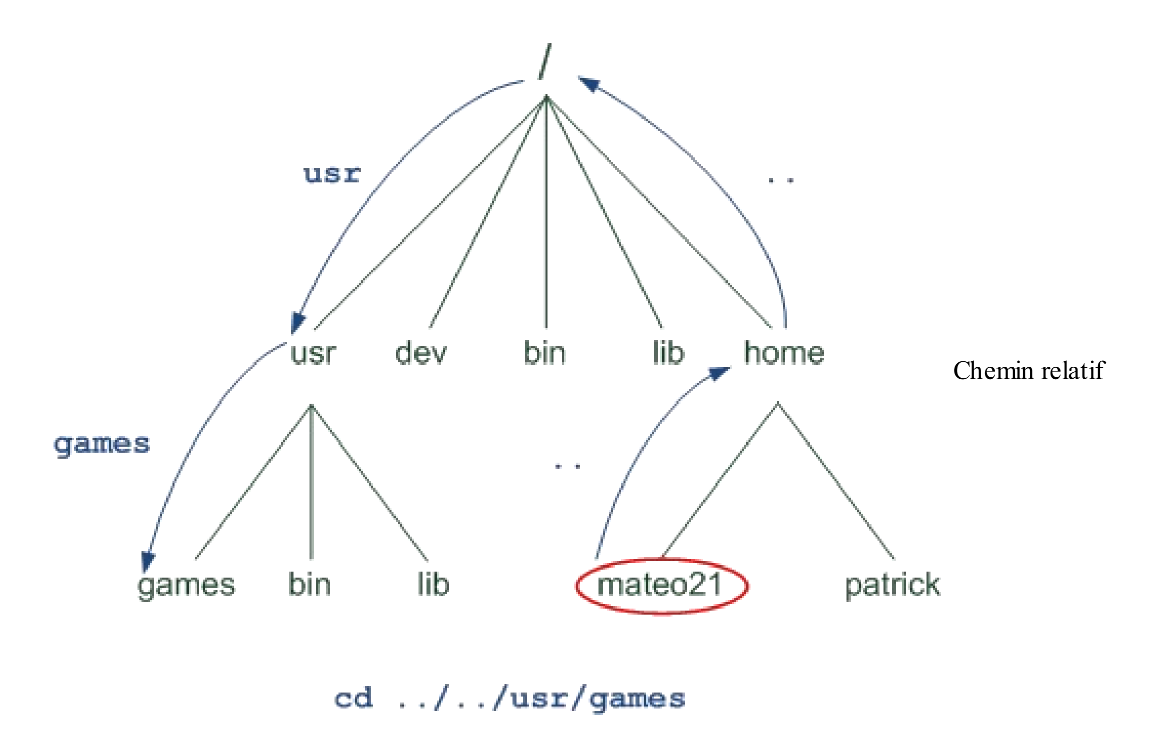
Le schéma suivante montre bien qu'on **part** de la racine / pour indiquer où on veut aller.

Si on avait voulu faire la même chose à coup de chemin relatif, il aurait fallu écrire :

Code : Console

```
mateo21@mateo21-desktop:~$ cd ../../usr/games/
mateo21@mateo21-desktop:/usr/games$
```

Ce qui signifie « **reviens en arrière (donc dans `/home`) puis reviens en arrière (donc dans `/`), puis va en avant dans `usr`, puis va en avant dans `games`** ». Voyez en figure suivante.



Ici, comme c'est un chemin relatif, on **part** du dossier dans lequel on se trouve (ici, c'est `/home/mateo21`) et on indique à la machine le chemin à suivre à partir de là pour aller dans le dossier qu'on veut.


Un chemin absolu est donc facile à reconnaître, car on part toujours de la racine /.
Un chemin relatif peut aussi s'avérer très pratique et plus court (ça dépend des cas).


Ce sera à vous de choisir à chaque fois comment vous voulez écrire votre chemin. Vous avez le choix.

Retour au répertoire home

Si vous voulez retourner dans votre répertoire `home` personnel, plusieurs solutions s'offrent à vous.

- **La brutale** : il suffit d'écrire le chemin absolu en entier. Cela donne :
  Code : Console
```
mateo21@mateo21-desktop:/usr/games$ cd /home/mateo21/
mateo21@mateo21-desktop:~$
```
- **La maligne** : plus court et plus pratique, vous pouvez utiliser l'alias ~ qui signifie la même chose. Cela donne :
  Code : Console
```
mateo21@mateo21-desktop:/usr/games$ cd ~
mateo21@mateo21-desktop:~$
```
- **La super maligne** : si vous ne mettez aucun paramètre à la commande `cd`, ça vous ramène aussi dans votre répertoire personnel.
  Code : Console
```
mateo21@mateo21-desktop:/usr/games$ cd
mateo21@mateo21-desktop:~$
```

Autocomplétion du chemin

Cette astuce est vitale ; si vous ne vous en servez pas, vous passez à côté d'une des plus importantes astuces de la console.

L'idée est simple : taper `cd /usr/games/trucbidule` c'est bien, mais c'est parfois un peu long de tout écrire. On a la flemme. Vous avez la flemme. Oui, vous. Alors vous allez justement demander à l'ordinateur de compléter le chemin tout seul !

L'autocomplétion de chemin fonctionne de la même manière que l'autocomplétion de commande qu'on a vue dans le chapitre précédent : avec la touche `Tab` (Tabulation). Faites le test avec moi. Commencez par vous placer dans `/usr` :

Code : Console
```
mateo21@mateo21-desktop:~$ cd /usr
mateo21@mateo21-desktop:/usr$
```

Tapez ensuite juste `cd ga`, puis appuyez sur `Tab`. C'est magique, le nom du dossier a été automatiquement complété !

Code : Console
```
mateo21@mateo21-desktop:/usr$ cd games/
```

Revenez maintenant dans `/usr` (en faisant `cd ..` par exemple) et essayez de taper juste `cd l`, puis faites `Tab`. Rien ne se passe : cela signifie que l'ordinateur n'a pas trouvé de dossier qui corresponde au début de votre recherche, ou alors qu'il y en plusieurs qui commencent par « l ». Faites à nouveau `Tab` :

Code : Console

```
mateo21@mateo21-desktop:/usr$ cd l
lib/    local/
mateo21@mateo21-desktop:/usr$ cd l
```

On vient de vous donner la liste des dossiers qui commencent par « l » ! Cela signifie qu'il faut préciser votre recherche parce que sinon, l'ordinateur ne peut pas deviner dans quel dossier vous voulez entrer. Ça tombe bien, la commande a été réécrite en dessous, vous n'avez plus qu'à ajouter une lettre plus précise : par exemple « o » pour que Linux devine que vous voulez aller dans le dossier `local`. Tapez donc « o », puis à nouveau `Tab`, et le nom sera complété !

Code : Console

```
mateo21@mateo21-desktop:/usr$ cd local/
```

Faites des tests pour vous entraîner à utiliser l'autocomplétion, c'est vraiment très important. Vous allez voir, c'est intuitif et vraiment pratique !

du: taille occupée par les dossiers

La commande « `du` », pour *Disk Usage* (utilisation du disque) vous donne des informations sur la taille qu'occupent les dossiers sur votre disque.

Placez-vous pour commencer dans `/usr/games`, et tapez `du` :

Code : Console

```
mateo21@mateo21-desktop:~$ cd /usr/games
mateo21@mateo21-desktop:/usr/games$ du
5732    .
```

Comme ce dossier ne contient pas de sous-dossier, la commande `du` nous renvoie la taille totale que font les fichiers contenus dans le dossier.

Si vous allez dans votre `home` en revanche, celui-ci contient beaucoup de sous-dossiers. Dans ce cas, la commande `du` va renvoyer la taille de chacun des sous-dossiers, puis la taille totale à la fin (« . ») :

Code : Console

```
mateo21@mateo21-desktop:/usr/games$ cd
mateo21@mateo21-desktop:~$ du
400     ./.Trash
4       ./.themes
32      ./.mozilla-thunderbird/8vyw6pqo.default/Mail/Local Folders
36      ./.mozilla-thunderbird/8vyw6pqo.default/Mail
12      ./.mozilla-thunderbird/8vyw6pqo.default/US
...
...
264     ./.jedit/jars
4       ./.jedit/macros
380     ./.jedit/settings-backup
856     ./.jedit
82484   .
```



J'ai volontairement coupé la liste car elle est très longue.

`-h` : la taille pour les humains

Ce qui est bien, c'est que les commandes reprennent souvent les mêmes paramètres. Ainsi, on avait vu `-h` pour `ls`, eh bien ce paramètre est le même pour avoir des tailles « humaines » avec `du` !

Code : Console

```
mateo21@mateo21-desktop:~$ du -h
400K    ./.Trash
4,0K    ./.themes
32K     ./.mozilla-thunderbird/8vyw6pqo.default/Mail/Local Folders
36K     ./.mozilla-thunderbird/8vyw6pqo.default/Mail
12K     ./.mozilla-thunderbird/8vyw6pqo.default/US
...
...
264K    ./.jedit/jars
4,0K    ./.jedit/macros
380K    ./.jedit/settings-backup
856K    ./.jedit
81M     .
```

Mon dossier `home` prend donc 81 Mo d'espace disque, son sous-dossier caché `.jedit` prend 856 Ko, etc.

`-a` : afficher la taille des dossiers ET des fichiers

Par défaut, `du` n'affiche que la taille des dossiers. Pour avoir aussi la taille des fichiers qu'ils contiennent, rajoutez l'option `-a` (*all*) :

Code : Console

```
mateo21@mateo21-desktop:~$ du -ah
...
8,0K    ./.jedit/settings-backup/abbrevs~5~
24K     ./.jedit/settings-backup/history~1~
8,0K    ./.jedit/settings-backup/abbrevs~4~
380K    ./.jedit/settings-backup
44K     ./.jedit/pluginMgr-Cached.xml.gz
856K    ./.jedit
81M     .
```

`-s` : avoir juste le grand total

Pour n'avoir que l'espace total occupé par le dossier et donc ne pas afficher le détail des sous-dossiers, utilisez `-s` (que je combine à `-h` pour plus de lisibilité) :

Code : Console

```
mateo21@mateo21-desktop:~$ du -sh
81M     .
```

Je vois ainsi que mon dossier `home` fait 81 Mo (rappel : le symbole point « . » signifie « **le dossier actuel** »).

En résumé

- Sous Linux, tout est organisé sous forme de fichiers. Il n'y a pas de lecteur du type `C:` comme sous Windows.
- Les dossiers sont imbriqués entre eux à partir du dossier parent principal `/`. On l'appelle la **racine**.
- Le dossier dans lequel les utilisateurs stockent leurs documents est `/home`. Si votre login est `patrick`, alors votre dossier personnel sera `/home/patrick`.
- La commande `pwd` permet de savoir en console dans quel dossier on se situe.
- `ls` affiche la liste des fichiers présents dans le dossier actuel.
- `cd` permet de changer de dossier.

 Manipuler les fichiers

Après avoir vu comment étaient organisés les fichiers sous Linux, nous allons apprendre à les manipuler !

Par exemple, comment faire pour afficher le contenu d'un fichier ?
Comment le déplacer, le copier, le supprimer ?

C'est donc un chapitre à la fois simple et riche qui vous attend, tout au long duquel vous allez apprendre beaucoup de nouvelles commandes basiques de Linux qu'il vous faut connaître absolument !

cat & less : afficher un fichier

Nous allons d'abord voir comment afficher le contenu d'un fichier. Il y a en gros deux commandes basiques sous Linux qui permettent de faire cela :

- `cat` ;
- `less`.

Aucune de ces commandes ne permet d'éditer un fichier, elles permettent juste de le **voir**. Nous étudierons l'édition plus tard, ça mérite au moins un chapitre entier.



Mais… pourquoi deux commandes pour afficher un fichier ? Une seule n'aurait pas suffi ?

En fait, chacune a ses spécificités ! Nous allons les voir dans le détail.

Pour nos exemples, nous allons travailler sur un fichier qui existe déjà : `syslog`. Il se trouve dans le dossier `/var/log`. Commencez par vous y rendre :

Code : Console

```
mateo21@mateo21-desktop:~$ cd /var/log
```

Ce dossier contient plusieurs fichiers de **log**, c'est-à-dire des fichiers qui gardent une trace de l'activité de votre ordinateur. Vous pouvez en faire la liste si vous le voulez, en tapant `ls` :

Code : Console

```
mateo21@mateo21-desktop:/var/log$ ls
acpid            daemon.log.0     kern.log.0               scrollkeeper.log.2
acpid.1.gz       daemon.log.1.gz  kern.log.1.gz            syslog
acpid.2.gz       daemon.log.2.gz  kern.log.2.gz            syslog.0
acpid.3.gz       daemon.log.3.gz  kern.log.3.gz            syslog.1.gz
acpid.4.gz       debug            lastlog                  syslog.2.gz
apparmor         debug.0          lpr.log                  syslog.3.gz
apport.log       debug.1.gz       mail.err                 syslog.4.gz
apport.log.1     debug.2.gz       mail.info                syslog.5.gz
apport.log.2.gz  debug.3.gz       mail.log                 syslog.6.gz
apport.log.3.gz  dist-upgrade     mail.warn                udev
apport.log.4.gz  dmesg            messages                 unattended-
upgrades
apport.log.5.gz  dmesg.0          messages.0               user.log
apt              dmesg.1.gz       messages.1.gz            user.log.0
auth.log         dmesg.2.gz       messages.2.gz            user.log.1.gz
auth.log.0       dmesg.3.gz       messages.3.gz            user.log.2.gz
auth.log.1.gz    dmesg.4.gz       news                     user.log.3.gz
auth.log.2.gz    dpkg.log         popularity-contest       uucp.log
auth.log.3.gz    dpkg.log.1       popularity-contest.0     wtmp
bittorrent       dpkg.log.2.gz    popularity-contest.1.gz  wtmp.1
boot             faillog          popularity-contest.2.gz  wvdialconf.log
bootstrap.log    fontconfig.log   popularity-contest.3.gz  Xorg.0.log
btmp             fsck             pycentral.log            Xorg.0.log.old
btmp.1           gdm              samba
```

```
cups            installer       scrollkeeper.log
daemon.log      kern.log        scrollkeeper.log.1
```

Le fichier sur lequel nous allons travailler, `syslog`, contient des informations de log de ce qui s'est passé récemment sur l'ensemble de votre ordinateur.



Vous noterez qu'il est fréquent de voir des fichiers sans extension sous Linux. Notre fichier s'appelle `syslog` tout court, et non pas `syslog.txt` ou `syslog.log` comme on pourrait avoir l'habitude de le voir sous Windows. Un fichier sans extension peut être ouvert et lu sans aucun problème comme n'importe quel autre fichier.

`cat` : afficher tout le fichier

La commande `cat` permet d'afficher tout le contenu d'un fichier dans la console d'un coup.
Il vous suffit d'indiquer en paramètre le nom du fichier que vous voulez afficher, en l'occurrence `syslog` :

Code : Console

```
mateo21@mateo21-desktop:/var/log$ cat syslog
Nov 14 00:44:23 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Job `cron.daily' terminated
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Normal exit (1 job run)
Nov 14 00:44:25 mateo21-desktop NetworkManager: <info>  eth1: link timed out.
Nov 14 00:44:51 mateo21-desktop NetworkManager: <info>  eth1: link timed out.
Nov 14 00:45:08 mateo21-
desktop NetworkManager: <debug> [1194997508.332093] nm_device_802_11_wireless_get_a
Nov 14 00:45:08 mateo21-
desktop NetworkManager: <info>  User Switch: /org/freedesktop/NetworkManager/Device
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Deactivating device eth1.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1): cancelli
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1) cancellat
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1): waiting
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1) cancellat
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1): cancelle
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  SUP: sending command 'DISAB
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  SUP: response was 'OK'
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  SUP: sending command 'AP_SC
Nov 14 00:45:16 mateo21-desktop NetworkManager: nm_act_request_get_ap: assertion `r
Nov 14 00:45:16 mateo21-desktop NetworkManager: nm_act_request_get_stage: assertion
Nov 14 00:45:16 mateo21-desktop NetworkManager: <info>  eth1: link timed out.
Nov 14 00:45:21 mateo21-desktop NetworkManager: nm_act_request_get_ap: assertion `r
Nov 14 00:45:21 mateo21-desktop NetworkManager: ap_is_auth_required: assertion `ap
Nov 14 00:45:21 mateo21-
desktop NetworkManager: <info>  Activation (eth1/wireless): association took too lo
Nov 14 00:45:21 mateo21-desktop NetworkManager: nm_dbus_get_user_key_for_network as
Nov 14 00:47:45 mateo21-desktop init: tty4 main process (4517) killed by TERM signa
Nov 14 00:47:45 mateo21-desktop init: tty5 main process (4518) killed by TERM signa
Nov 14 00:47:45 mateo21-desktop init: tty2 main process (4520) killed by TERM signa
Nov 14 00:47:45 mateo21-desktop init: tty3 main process (4522) killed by TERM signa
Nov 14 00:47:45 mateo21-desktop init: tty1 main process (4524) killed by TERM signa
Nov 14 00:47:45 mateo21-desktop init: tty6 main process (4525) killed by TERM signa
Nov 14 00:47:46 mateo21-desktop avahi-daemon[5390]: Got SIGTERM, quitting.
Nov 14 00:47:48 mateo21-desktop exiting on signal 15
Nov 14 00:48:42 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.

...
```

Comme le fichier est très gros (il fait plusieurs centaines de lignes), je n'ai pas copié tout ce qui s'est affiché dans ma console. Ne vous étonnez pas si vous voyez tout s'afficher d'un coup : c'est normal, c'est le but. La commande `cat` vous envoie tout le fichier à la figure. Elle est plus adaptée lorsque l'on travaille sur de petits fichiers que sur des gros, car dans un cas comme celui-

là, on n'a pas le temps de lire tout ce qui s'affiche à l'écran.

Il y a peu de paramètres vraiment intéressants à utiliser avec la commande `cat`, car c'est une commande somme toute très basique. On notera quand même le paramètre `-n` qui permet d'afficher les numéros de ligne :

Code : Console

```
mateo21@mateo21-desktop:/var/log$ cat -n syslog
     1        Nov 14 00:44:23 mateo21-
desktop syslogd 1.4.1#21ubuntu3: restart.
     2        Nov 14 00:44:23 mateo21-
desktop anacron[6725]: Job `cron.daily' terminated
     3        Nov 14 00:44:23 mateo21-
desktop anacron[6725]:         Normal exit (1 job run)
     4        Nov 14 00:44:25 mateo21-
desktop NetworkManager: <info>  eth1: link timed out.
     5        Nov 14 00:44:51 mateo21-
desktop NetworkManager: <info>  eth1: link timed out.
     6        Nov 14 00:45:08 mateo21-
desktop NetworkManager: <debug> [1194997508.332093]

...
```

less : afficher le fichier page par page

La commande `cat` est rapide. Trop rapide. Tout le fichier est lu et affiché d'un coup dans la console, ce qui fait que l'on n'a pas le temps de le lire s'il est très gros.

C'est là qu'une autre commande comme `less` devient vraiment indispensable. La grosse différence entre `less` et `cat`, c'est que `less` affiche progressivement le contenu du fichier, page par page. Ça vous laisse le temps de le lire dans la console. :-)



Notez qu'il existe aussi une commande très proche : `more`. Pour faire simple, la différence entre `more` et `less` c'est que `more` est vieux et possède peu de fonctionnalités, tandis que `less` est beaucoup plus puissant et rapide. Bref, utilisez `less`, mais si vous voyez un jour quelqu'un utiliser `more`, ne soyez pas surpris.

Comment ça marche ? Eh bien la commande est très simple : `less nomdufichier`.

Code : Console

```
mateo21@mateo21-desktop:/var/log$ less syslog
Nov 14 00:44:23 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Job `cron.daily' terminated
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Normal exit (1 job run)
Nov 14 00:44:25 mateo21-desktop NetworkManager: <info>  eth1: link timed out.
Nov 14 00:44:51 mateo21-desktop NetworkManager: <info>  eth1: link timed out.
Nov 14 00:45:08 mateo21-
desktop NetworkManager: <debug> [1194997508.332093] nm_device_802_11_wireless_get_a
Nov 14 00:45:08 mateo21-
desktop NetworkManager: <info>  User Switch: /org/freedesktop/NetworkManager/Device
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Deactivating device eth1.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1): cancelli
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1) cancellat
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1): waiting
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1) cancellat
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1): cancelle
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  SUP: sending command 'DISAB
```

Ce qui est intéressant pour nous ici, c'est que la commande `less` a arrêté la lecture du fichier au bout de quelques lignes (la taille d'un écran de console). Cela vous laisse le temps de lire le début du fichier.

On n'a lu pour le moment que les toutes premières lignes du fichier.



Et comment lire la suite ?

Il y a quelques raccourcis clavier à connaître. ;-)

Les raccourcis basiques indispensables

Commençons par les quelques raccourcis clavier les plus indispensables, à connaître absolument.

- `Espace` : affiche la suite du fichier. La touche `Espace` fait défiler le fichier vers le bas d'un « écran » de console. C'est celle que j'utilise le plus souvent.
- `Entrée` : affiche la ligne suivante. Cela permet donc de faire défiler le fichier vers le bas ligne par ligne.
  Vous pouvez aussi utiliser la touche `Flèche vers le bas`.
- `d` : affiche les onze lignes suivantes (soit une moitié d'écran). C'est un peu l'intermédiaire entre `Espace` (tout un écran) et `Entrée` (une seule ligne).
- `b` : retourne en arrière d'un écran.
  Vous pouvez aussi appuyer sur la touche `Page Up`.
- `y` : retourne d'une ligne en arrière.
  Vous pouvez aussi appuyer sur la touche `Flèche vers le haut`.
- `u` : retourne en arrière d'une moitié d'écran (onze lignes).
- `q` : arrête la lecture du fichier. Cela met fin à la commande `less`.



La casse des caractères est importante. Ainsi, si je vous dis qu'il faut appuyer sur la touche « d », ce n'est pas un « D » majuscule (si vous essayez, vous verrez que ça ne fonctionne pas). Sous Linux, on fait souvent la différence entre majuscules et minuscules : souvenez-vous-en !

Si on tape `Espace`, on avance donc d'un écran dans le fichier :

Code : Console

```
Nov 14 00:47:45 mateo21-
desktop init: tty4 main process (4517) killed by TERM signal
Nov 14 00:47:45 mateo21-
desktop init: tty5 main process (4518) killed by TERM signal
Nov 14 00:47:45 mateo21-
desktop init: tty2 main process (4520) killed by TERM signal
Nov 14 00:47:45 mateo21-
desktop init: tty3 main process (4522) killed by TERM signal
Nov 14 00:47:45 mateo21-
desktop init: tty1 main process (4524) killed by TERM signal
Nov 14 00:47:45 mateo21-
desktop init: tty6 main process (4525) killed by TERM signal
Nov 14 00:47:46 mateo21-desktop avahi-
daemon[5390]: Got SIGTERM, quitting.
Nov 14 00:47:48 mateo21-desktop exiting on signal 15
Nov 14 00:48:42 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
Nov 14 00:48:42 mateo21-desktop kernel: Inspecting /boot/System.map-
2.6.22-14-generic
Nov 14 00:48:42 mateo21-
desktop kernel: Loaded 25445 symbols from /boot/System.map-2.6.22-14-
generic.
Nov 14 00:48:42 mateo21-
desktop kernel: Symbols match kernel version 2.6.22.
Nov 14 00:48:42 mateo21-desktop kernel: No module symbols loaded -
 kernel modules not enabled.
Nov 14 00:48:42 mateo21-
desktop kernel: [    0.000000] Linux version 2.6.22-14-
generic (buildd@palmer) (gcc version 4.1.3 20070929 (prerelease)
```

```
:
```

Quelques raccourcis plus avancés

Ce ne sont pas des raccourcis que l'on utilise tous les jours, mais ça vaut le coup de savoir qu'ils existent. :-)

- `=` : indique où vous en êtes dans le fichier (numéro des lignes affichées et pourcentage).
- `h` : affiche l'aide (toutes les commandes que je vous apprends ici, je les tire de là). Tapez `q` pour sortir de l'aide.
- `/` : tapez `/` suivi du texte que vous recherchez pour lancer le mode recherche. Faites `Entrée` pour valider. Pour ceux qui savent s'en servir, sachez que les expressions régulières sont acceptées. Je ne vais pas vous faire un cours sur les expressions régulières ici, ce serait trop long, mais il y en a un dans mon cours sur le PHP *Concevez votre site Web avec PHP et MySQL* dans la même collection.
- `n` : après avoir fait une recherche avec `/`, la touche `n` vous permet d'aller à la prochaine occurrence de votre recherche. C'est un peu comme si vous cliquiez sur le bouton « Résultat suivant ».
- `N` : pareil que `n`, mais pour revenir en arrière.

Comme vous le voyez, la commande `less` est très riche. On peut utiliser beaucoup de touches différentes pour se déplacer dans le fichier.
Prenez le temps de vous familiariser avec : c'est un peu perturbant au début, mais lorsque vous aurez appris à vous en servir, vous aurez déjà fait un grand pas en avant… et puis ça vous sera très utile plus tard, croyez-moi. ;-)

head & tail : afficher le début et la fin d'un fichier


Quoiii ? Encore des commandes pour lire un fichier ?

Eh oui.
Et figurez-vous que celles-là aussi valent le coup d'être connues. Comme quoi on en fait des commandes, rien que pour lire un fichier !

Ces deux commandes sont un peu à l'opposé l'une de l'autre : la première permet d'afficher le début du fichier, la seconde permet d'afficher la fin.

head : afficher le début du fichier

La commande `head` (« tête » en anglais) affiche seulement les premières lignes du fichier. Elle ne permet pas de se déplacer dans le fichier comme `less`, mais juste de récupérer les premières lignes.

Code : Console

```
mateo21@mateo21-desktop:/var/log$ head syslog
Nov 14 00:44:23 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Job `cron.daily' terminated
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Normal exit (1 job run)
Nov 14 00:44:25 mateo21-desktop NetworkManager: <info>  eth1: link timed out.
Nov 14 00:44:51 mateo21-desktop NetworkManager: <info>  eth1: link timed out.
Nov 14 00:45:08 mateo21-
desktop NetworkManager: <debug> [1194997508.332093] nm_device_802_11_wireless_get_a
Nov 14 00:45:08 mateo21-
desktop NetworkManager: <info>  User Switch: /org/freedesktop/NetworkManager/Device
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Deactivating device eth1.
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1): cancelli
Nov 14 00:45:08 mateo21-desktop NetworkManager: <info>  Activation (eth1) cancellat
```

Si vous avez juste besoin de récupérer les premières lignes d'un fichier, `head` est donc la commande qu'il vous faut. Simple, net, efficace.

Comment ? Vous voulez des paramètres ?
Je n'en ai pas beaucoup à vous offrir, mais celui-là au moins est à connaître : `-n`, suivi d'un nombre. Il permet d'afficher le nombre de lignes que vous voulez. Par exemple, si vous ne voulez que les trois premières lignes, tapez :

Code : Console

```
mateo21@mateo21-desktop:/var/log$ head -n 3 syslog
Nov 14 00:44:23 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.
Nov 14 00:44:23 mateo21-
desktop anacron[6725]: Job `cron.daily' terminated
Nov 14 00:44:23 mateo21-desktop anacron[6725]: Normal exit (1 job run)
```

Et voilà le travail !

`tail` : afficher la fin du fichier

Très intéressante aussi (voire même plus), la commande `tail` vous renvoie la fin du fichier, donc les dernières lignes.

Code : Console

```
mateo21@mateo21-desktop:/var/log$ tail syslog
Nov 14 22:42:10 mateo21-desktop NetworkManager: <info> Clearing nscd hosts cache.
Nov 14 22:42:10 mateo21-desktop NetworkManager: <WARN>  nm_spawn_process(): nm_spaw
i hosts'): could not spawn process. (Failed to execute child process "/usr/sbin/nsc
Nov 14 22:42:10 mateo21-desktop NetworkManager: <info>  Activation (eth1) Finish ha
Nov 14 22:42:10 mateo21-
desktop NetworkManager: <info>  Activation (eth1) Stage 5 of 5 (IP Configure Commit
Nov 14 22:42:10 mateo21-desktop NetworkManager: <info>  Activation (eth1) successfu
Nov 14 22:41:57 mateo21-desktop ntpdate[8422]: step time server 91.189.94.4 offset
Nov 14 22:41:59 mateo21-desktop avahi-
daemon[5385]: Registering new address record for fe80::219:d2ff:fe61:900a on eth1.*
Nov 14 22:42:08 mateo21-desktop kernel: [ 7870.160000] eth1: no IPv6 routers presen
Nov 14 23:11:26 mateo21-desktop -- MARK --
Nov 14 23:17:01 mateo21-desktop /USR/SBIN/CRON[8515]: (root) CMD (   cd / && run-pa
```

On peut là encore utiliser `-n` suivi d'un nombre pour afficher les x dernières lignes :

Code : Console

```
mateo21@mateo21-desktop:/var/log$ tail -n 3 syslog
Nov 14 22:42:08 mateo21-
desktop kernel: [ 7870.160000] eth1: no IPv6 routers present
Nov 14 23:11:26 mateo21-desktop -- MARK --
Nov 14 23:17:01 mateo21-
desktop /USR/SBIN/CRON[8515]: (root) CMD (   cd / && run-parts --
report /etc/cron.hourly)
```

Mais ce n'est pas tout ! Il y a un autre paramètre à côté duquel vous ne pouvez pas passer : `-f` (f pour *follow*, « suivre » en anglais).

Ce paramètre magique ordonne à `tail` de « suivre » la fin du fichier au fur et à mesure de son évolution.
C'est extrêmement utile pour suivre un fichier de log qui évolue souvent. Vous pouvez tester sur `syslog` par exemple :

Code : Console

```
mateo21@mateo21-desktop:/var/log$ tail -f syslog
Nov 14 23:11:26 mateo21-desktop -- MARK --
Nov 14 23:17:01 mateo21-desktop /USR/SBIN/CRON[8515]: (root) CMD (   cd / && run-pa
report /etc/cron.hourly)
Nov 14 23:27:52 mateo21-
desktop kernel: [10614.344000] ata2.00: exception Emask 0x0 SAct 0x0 SErr 0x0 actic
Nov 14 23:27:52 mateo21-
desktop kernel: [10614.344000] ata2.00: cmd a0/00:00:00:00:20/00:00:00:00:00/a0 tag
Nov 14 23:27:52 mateo21-
desktop kernel: [10614.344000]         res 40/00:03:00:00:00/00:00:00:00:00/a0 Ema
Nov 14 23:27:57 mateo21-
desktop kernel: [10619.388000] ata2: port is slow to respond, please be patient (St
Nov 14 23:28:02 mateo21-desktop kernel: [10624.392000] ata2: device not ready (errn
16), forcing hardreset
Nov 14 23:28:02 mateo21-desktop kernel: [10624.392000] ata2: soft resetting port
Nov 14 23:28:02 mateo21-desktop kernel: [10624.928000] ata2.00: configured for UDMA
Nov 14 23:28:02 mateo21-desktop kernel: [10624.928000] ata2: EH complete
```

Le problème de `syslog` c'est qu'il n'évolue pas forcément toutes les secondes. Mais si vous êtes patients et que vous regardez votre console, vous devriez le voir écrire de nouvelles lignes sous vos yeux au bout d'un moment.

Faites `Ctrl + C` (`Ctrl` et `C` en même temps) pour arrêter la commande `tail`.



À connaître : la combinaison de touches `Ctrl + C` est utilisable dans la plupart des programmes console pour demander leur arrêt. C'est un peu l'équivalent du `Alt + F4` de Windows.

Pour tout vous dire, `tail -f` est une de mes commandes préférées sous Linux. C'est un bon moyen de surveiller ce qui se passe en temps réel sur un ordinateur (si vous êtes assez rapides pour suivre).

Par exemple, les logs Apache du Site du Zéro permettent de voir en temps réel qui se connecte sur le site, avec quelle IP, quel fichier a été chargé, à quelle heure, etc.
Aux heures d'affluence du site, ce fichier évolue tellement vite qu'il est pratiquement impossible de le suivre pour un humain.

Je vous ai fait une petite vidéo d'un `tail -f` en action pour que vous vous rendiez compte de la chose. Elle est accessible par téléchargement (380 Ko) :

Notez que par défaut, `tail -f` recherche les nouveaux changements dans le fichier toutes les secondes. Si vous voulez, vous pouvez rajouter le paramètre `-s` suivi d'un nombre. Par exemple, `tail -f -s 3 syslog` recherchera les changements toutes les trois secondes (plutôt que toutes les secondes). Les nombres décimaux sont acceptés, à condition d'utiliser le point « . » à la place de la virgule.

touch & mkdir : créer des fichiers et dossiers

Assez lu de fichiers, maintenant voyons voir comment on les crée !

Nous allons d'abord voir comment créer un fichier, puis comment créer un dossier, car ce n'est pas la même commande…

touch : créer un fichier

En fait, il n'existe aucune commande spécialement faite pour créer un fichier vide sous Linux (ce n'est pas très utile). En général, on se contente d'ouvrir un éditeur de texte et d'enregistrer, ce qui provoque la création d'un fichier comme sous Windows.

La commande `touch` est à la base faite pour modifier la date de dernière modification d'un fichier. D'où son nom : on « touche » le fichier pour faire croire à l'ordinateur qu'on vient de le modifier alors que l'on n'a rien changé. Ça peut se révéler utile dans certains cas précis qu'on ne verra pas ici.

L'intérêt de `touch` pour nous dans ce chapitre, c'est que si le fichier n'existe pas, il sera créé ! On peut donc **aussi** utiliser `touch` pour créer des fichiers, même s'il n'a pas vraiment été fait pour ça à la base.

La commande attend un paramètre : le nom du fichier à créer.
Commencez par vous rendre dans votre dossier personnel ; ce n'est pas une bonne idée de mettre le bazar dans `/var/log`, le

dossier personnel est là pour ça.
Si vous vous souvenez bien, il suffit de taper `cd` :

Code : Console

```
mateo21@mateo21-desktop:/var/log$ cd
mateo21@mateo21-desktop:~$
```

Pour le moment, mon dossier personnel ne contient que des sous-dossiers :

Code : Console

```
mateo21@mateo21-desktop:~$ ls -F
Desktop/    Examples@  Images/  Modèles/  Public/  Vidéos/
Documents/  images/    log/     Musique/  tutos/
```

Créons un fichier appelé `fichierbidon` :

Code : Console

```
mateo21@mateo21-desktop:~$ touch fichierbidon
mateo21@mateo21-desktop:~$ ls -F
Desktop/    Examples@     images/  log/      Musique/  tutos/
Documents/  fichierbidon  Images/  Modèles/  Public/   Vidéos/
```

La commande `ls -F` que j'ai tapée ensuite le montre, un fichier appelé `fichierbidon` (sans extension) a été créé. Bien entendu, vous pouvez créer un fichier de l'extension que vous voulez :

Code : Console

```
mateo21@mateo21-desktop:~$ touch autrefichierbidon.txt
mateo21@mateo21-desktop:~$ ls -F
autrefichierbidon.txt  Examples@     Images/   Musique/  Vidéos/
Desktop/               fichierbidon  log/      Public/
Documents/             images/       Modèles/  tutos/
```

Autre information intéressante : vous pouvez créer plusieurs fichiers en une seule commande. Il vous suffit de les lister l'un après l'autre, séparés par des espaces.
Ainsi, on aurait pu créer nos deux fichiers comme ceci :

Code : Console

```
touch fichierbidon autrefichierbidon.txt
```



Et si je veux que mon fichier contienne un espace, je fais comment ?

Entourez-le de guillemets !

Code : Console

```
touch "Fichier bidon"
```

mkdir : créer un dossier

La commande `mkdir`, elle, est faite pour créer un dossier. Elle fonctionne de la même manière que `touch`.

Code : Console

```
mkdir mondossier
```

On peut créer deux dossiers (ou plus !) en même temps en les séparant là aussi par des espaces :

Code : Console

```
mkdir mondossier autredossier
```

Si vous faites un `ls`, vous verrez que les dossiers ont bien été créés. :-)

Il y a un paramètre utile avec `mkdir` : `-p`. Il sert à créer tous les dossiers intermédiaires. Par exemple :

Code : Console

```
mkdir -p animaux/vertebres/chat
```

… créera le dossier `animaux`, puis à l'intérieur le sous-dossier `vertebres`, puis à l'intérieur encore le sous-dossier `chat` !

cp & mv : copier et déplacer un fichier

Parmi les opérations de base que l'on veut pouvoir faire avec les fichiers, il y a la copie et le déplacement de fichier. C'est un peu le genre de chose que l'on fait tous les jours, il est donc important de savoir s'en servir.

cp : copier un fichier

La commande `cp` (abréviation de *CoPy*, « copier » en anglais) vous permet comme son nom l'indique de copier un fichier… mais aussi de copier plusieurs fichiers à la fois, et même de copier des dossiers !

Si on essayait de copier le fichier `fichierbidon` qu'on a créé tout à l'heure ?
Ça fonctionne comme ceci :

Code : Console

```
cp fichierbidon fichiercopie
```

Le premier paramètre est le nom du fichier à copier, le second le nom de la copie du fichier à créer.
En faisant cela, on aura donc deux fichiers identiques dans le même répertoire : `fichierbidon` et `fichiercopie`.



N'oubliez pas d'utiliser l'autocomplétion avec la touche `Tab` ! Lorsque vous avez écrit `cp fic`, tapez `Tab`, et `fichierbidon` devrait se compléter tout seul !

Copier dans un autre dossier

On n'est pas obligé de copier le fichier dans le même dossier, bien sûr. On peut très bien utiliser le système de répertoires relatifs et absolus qu'on a vu dans le chapitre précédent.
Par exemple, si je veux copier `fichierbidon` dans le sous-dossier `mondossier` que j'ai créé tout à l'heure :

Code : Console

```
cp fichierbidon mondossier/
```

Le fichier `fichierbidon` sera copié dans `mondossier` sous le même nom.



Notez que mettre le `/` à la fin n'est pas obligatoire. Si vous le voyez là, c'est parce que l'autocomplétion me l'a automatiquement ajouté lorsque j'ai appuyé sur `Tab`. Ehhh oui, je suis tellement flemmard que je n'écris même pas `mondossier` en entier, j'écris juste `mon` suivi de `Tab`, et hop c'est écrit en entier ! Ça va beaucoup plus vite lorsqu'on prend ce réflexe.

Si vous voulez copier `fichierbidon` dans `mondossier` sous un autre nom, faites comme ceci :

Code : Console

```
cp fichierbidon mondossier/fichiercopie
```

Avec cette commande, on aura créé une copie de `fichierbidon` dans `mondossier` sous le nom `fichiercopie` !

Enfin là, j'utilise des répertoires relatifs, mais je peux aussi écrire un répertoire en absolu :

Code : Console

```
cp fichierbidon /var/log/
```

… copiera `fichierbidon` dans le dossier `/var/log`.

Copier des dossiers

Avec l'option `-R` (un « R » majuscule !), vous pouvez copier un dossier, ainsi que tous les sous-dossiers et fichiers qu'il contient !

Tout à l'heure, on a créé un dossier `animaux` qui contenait un autre dossier `vertebres`, qui lui-même contenait le dossier `chat`. Si vous tapez cette commande :

Code : Console

```
cp -R animaux autresanimaux
```

… cela aura pour effet de copier `animaux` ainsi que tous ses sous-dossiers sous le nom `autresanimaux`.
Faites des `ls` après pour vérifier que les sous-dossiers sont bien là et que je ne vous mène pas en bateau !

*Utiliser le joker **

Le symbole `*` est appelé *joker*, ou encore ***wildcard*** en anglais sous Linux.
Il vous permet de copier par exemple tous les fichiers image `.jpg` dans un sous-dossier :

Code : Console

```
cp *.jpg mondossier/
```

Vous pouvez aussi vous en servir pour copier tous les fichiers dont le nom commence par « so » :

Code : Console

```
cp so* mondossier/
```

Le joker est un atout très puissant, n'hésitez pas à l'utiliser !
C'est avec des outils comme le joker que la console deviendra pour vous progressivement plus puissante que l'explorateur de fichiers que vous manipulez à la souris.

`mv` : déplacer un fichier

Très proche de `cp`, la commande `mv` (*MoVe*, « déplacer » en anglais) a en fait deux utilités :

- déplacer un fichier (ou un dossier) ;
- renommer un fichier (ou un dossier).

Vous allez comprendre pourquoi.

Déplacer un fichier

La commande `mv` s'utilise pratiquement comme `cp` :

Code : Console

```
mv fichierbidon mondossier/
```

Au lieu de copier `fichierbidon` dans `mondossier` comme on l'a fait tout à l'heure, ici on a juste déplacé le fichier. Il n'existe plus dans son dossier d'origine.

Vous pouvez déplacer des dossiers aussi simplement :

Code : Console

```
mv animaux/ mondossier/
```

… déplacera le dossier `animaux` (et tous ses sous-dossiers) dans `mondossier`.

Vous pouvez aussi utiliser les jokers :

Code : Console

```
mv *.jpg mondossier/
```

Renommer un fichier

La commande `mv` permet de faire quelque chose d'assez étonnant : renommer un fichier. En effet, il n'existe pas de commande spéciale pour renommer un fichier en console sous Linux, c'est la commande `mv` qui est utilisée pour ça.

Par exemple :

Code : Console

```
mv fichierbidon superfichier
```

… renommera `fichierbidon` en `superfichier`. Après cette commande, `fichierbidon` n'existe plus, il a été renommé.

Déplacer et renommer un fichier à la fois

Vous pouvez aussi déplacer `fichierbidon` dans `mondossier` tout en lui affectant un nouveau nom :

Code : Console

```
mv fichierbidon mondossier/superfichier
```

Et voilà le travail !

Je vous conseille **fortement** de vous entraîner à utiliser `cp` et `mv` dans tous les sens : avec ou sans joker, en déplaçant, renommant des dossiers, en déplaçant / renommant à la fois, en utilisant des chemins relatifs et absolus, etc.
C'est assez intuitif normalement, mais il faut pratiquer et pas seulement se contenter de lire ce que j'écris pour que ça rentre.



N'oubliez pas d'utiliser l'autocomplétion de fichiers et dossiers avec la touche `Tab` ; si vous ne le faites pas dès maintenant, vous perdrez du temps et vous trouverez la console nulle alors que vous devriez la trouver géniale.
Autre chose : le symbole `..` signifie « dossier précédent », et `.` signifie « dossier dans lequel je me trouve ». Vous pourriez en avoir besoin lorsque vous copiez ou déplacez un fichier.

Si vous avez la tête qui tourne à force de copier et déplacer des fichiers dans des dossiers, c'est normal. Ça commence à devenir un beau bazar dans vos dossiers d'ailleurs, non ?
Il est temps de faire un peu de ménage avec la commande permettant de **supprimer** : `rm` !

rm : supprimer des fichiers et dossiers

On attaque la commande qui fâche : `rm`.
Pourquoi est-ce qu'elle fâche ? Parce qu'il n'existe pas de corbeille dans la console de Linux : le fichier est directement supprimé sans possibilité de récupération !

`rm` : supprimer un fichier

La commande `rm` (pour *ReMove*, « supprimer » en anglais) peut supprimer un fichier, plusieurs fichiers, des dossiers, voire même votre ordinateur entier si vous le voulez.

Il faut donc l'utiliser avec précaution. Commençons par des choses simples, supprimons ce `fichierbidon` :

Code : Console

```
rm fichierbidon
```

Normalement, on ne vous demande pas de confirmation, on ne vous affiche rien. Le fichier est supprimé sans autre forme d'avertissement. Brutal, hein ?

Vous pouvez aussi supprimer plusieurs fichiers en séparant leurs noms par des espaces :

Code : Console

```
rm fichierbidon fichiercopie
```

-i : demander confirmation

La commande `-i` permet de vous demander une confirmation pour chacun des fichiers :

Code : Console

```
mateo21@mateo21-desktop:~$ rm -i fichierbidon
rm: détruire fichier régulier vide `fichierbidon'?
```

Lorsqu'on vous demande une confirmation de type oui/non comme ici, vous devez répondre par une lettre :

- **o** : signifie « Oui ». Sur certains systèmes anglais, il faudra peut-être utiliser `y` de *Yes* ;
- **n** : signifie « Non ».

Tapez ensuite sur `Entrée` pour valider.

-f : forcer la suppression, quoi qu'il arrive

`-f`, c'est un peu le contraire de `-i` : c'est le mode des gros bourrins.
Ce paramètre force la suppression, ne demande pas de confirmation, même s'il y a un problème potentiel.

En raison des risques que cela comporte, utilisez-le aussi rarement que possible.

Code : Console

```
rm -f fichierbidon
```

-v : dis-moi ce que tu fais, petit cachotier

Le paramètre `-v` (*Verbose*, verbeux en anglais, c'est-à-dire « parler beaucoup ») est un paramètre que l'on retrouve dans beaucoup de commandes sous Linux. Il permet de demander à la commande de dire ce qu'elle est en train de faire.

Comme vous l'avez vu, par défaut la commande `rm` est silencieuse. Si vous supprimez de très nombreux fichiers, ça peut prendre du temps. Pour éviter que vous vous impatientiez, pensez à utiliser `-v` :

Code : Console

```
mateo21@mateo21-desktop:~$ rm -v fichierbidon fichiercopie
détruit `fichierbidon'
détruit `fichiercopie'
```

Vous voyez au fur et à mesure de l'avancement ce qui est en train d'être fait. Très pratique !

`-r` : supprimer un dossier et son contenu

Le paramètre `-r` peut être utilisé pour supprimer un dossier (au lieu d'un fichier) ainsi que tout ce qu'il contient : fichiers et dossiers !

C'est un paramètre assez dangereux, faites donc bien attention de l'utiliser sur un dossier dont vous ne voulez vraiment plus, car tout va disparaître à l'intérieur :

Code : Console

```
rm -r animaux/
```

… supprime le dossier `animaux` ainsi que tout ce qu'il contenait (sous-dossiers `vertebres` et `chat`).



Notez qu'il existe aussi la commande `rmdir`. La grosse différence avec `rm -r`, c'est que `rmdir` ne peut supprimer un dossier que s'il est vide ! Il faudra y avoir fait le ménage auparavant.

`rm` et le joker de la mort (qui tue)

Bon, vous êtes grands, je crois que le moment est venu de vous révéler un terrible secret : ~~les enfants ne naissent pas dans les choux~~.
Euh pardon, je voulais dire : la commande `rm` est vraiment dangereuse. Très dangereuse. Vous pouvez potentiellement bousiller tout votre système avec !



Je vais vous montrer quelque chose d'horrible, d'affreux, d'interdit aux moins de 18 ans, bref vous m'avez compris, le truc à ne faire sous aucun prétexte, même pas en cauchemar.

Code : Console

```
NON NON NON NE FAITES JAMAIS CA !!! => rm -rf /*
```

Je me suis permis de mettre du texte avant pour vous éviter la tentation de recopier bêtement la commande pour « rigoler », pour « voir ce que ça fait ». Je vais vous l'expliquer dans le détail, parce que c'est quand même **l'erreur n° 1** à ne pas faire sous Linux.

- `rm` : commande la suppression ;
- `-r` : supprime de manière récursive tous les fichiers et dossiers ;
- `-f` : force la suppression sans demander la moindre confirmation ;
- `/*` : supprime tous les fichiers et dossiers qui se trouvent à la racine (`/`) quel que soit leur nom (joker `*`).

En clair, cette commande supprime tout votre disque dur depuis la racine, sous-dossiers compris, et ne demande aucune confirmation. Aucune possibilité de récupération, votre PC est foutu. Vous êtes bons pour une réinstallation de Linux, **et aussi de Windows** si la partition de Windows était accessible depuis Linux.



Mais ils sont bêtes les gens qui ont créé cette commande ! Pourquoi autoriser de faire une chose aussi risquée ?

En fait, il y a plusieurs mécanismes de protection. On en apprendra plus dans le prochain chapitre (qui traitera des utilisateurs et

de leurs droits).
Par exemple, les fichiers à la racine ne vous « appartiennent » pas, ils appartiennent au superutilisateur « root ». Moi je me suis loggé en tant que `mateo21`, je n'ai donc théoriquement pas le droit de supprimer ces fichiers. La suppression sera refusée.

Seulement, pour peu que vous soyez loggés en tant que « root » (on verra comment le faire dans le chapitre suivant), vous aurez le droit de le faire, et là **plus rien ne vous arrêtera** !

On apprendra plus tard comment utiliser les alias de commande pour éviter qu'une commande aussi dangereuse ne s'exécute. En attendant, ne jouez pas avec le feu, car vous y perdriez les mains, les pieds, la tête et tout ce qui va avec.

Le joker reste quand même très utile, mais lorsque vous l'utilisez avec `rm`, triplez d'attention.
Par exemple :

Code : Console

```
rm -rf *
```

… supprime tous les fichiers et sous-dossiers du dossier dans lequel je me trouve. Il m'arrive de l'utiliser, d'en avoir besoin, mais à chaque fois je fais très très attention à ce qu'il n'y ait plus rien dans ce dossier (et dans les sous-dossiers) qui m'intéresse.
Comme vous pouvez le voir, il n'y a qu'un seul caractère de différence (le `/`) avec la commande de la mort que je vous ai montrée un peu plus haut.

Une erreur est vite arrivée. J'ignore combien de gens se sont pendus après avoir exécuté cette commande, mais ça méritait au moins un **GROS** avertissement !

ln : créer des liens entre fichiers

Bien qu'un peu moins courante, la commande `ln` vous sera certainement utile un jour ou l'autre. Elle permet de créer des liens entre des fichiers, c'est-à-dire (pour employer des mots que vous connaissez) qu'elle permet de **créer des raccourcis**.

Ces « raccourcis », qu'on appelle des **liens** sous Linux, sont un peu plus complexes que ceux que vous avez l'habitude de voir sous Windows. En effet, on peut créer deux types de liens :

- des liens **physiques** ;
- des liens **symboliques**.

Ces deux types ne fonctionnent pas de la même manière. Pour comprendre ce qui les différencie, il faut savoir comment un OS tel que Linux gère les fichiers sur le disque dur.
Allons, allons, ne faites pas cette tête-là, un peu de théorie sur le fonctionnement des OS, c'est toujours très intéressant ! :-)

Le stockage des fichiers

Sur le disque dur, chaque fichier est grosso-modo séparé en deux parties :

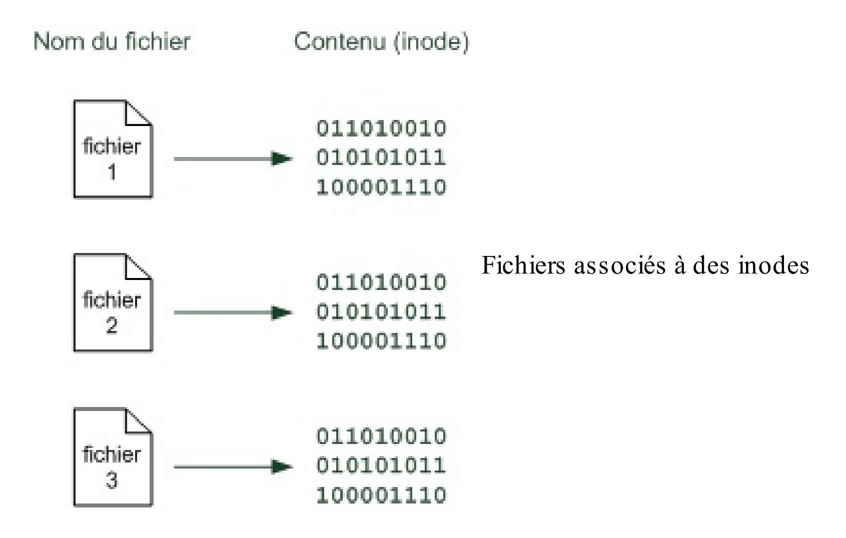
- son nom ;
- son contenu.

Vous avez bien lu : la liste des noms de fichiers est stockée à un autre endroit que leur contenu. Cette séparation aide Linux à s'organiser.



Je simplifie ici volontairement les choses. En pratique, c'est (toujours) un peu plus compliqué. Il y en fait trois parties : le nom, les informations de gestion (droits d'accès) et le contenu. Mais nous allons faire simple car notre but est juste de comprendre l'idée générale du fonctionnement.

Chaque contenu de fichier se voit attribuer un numéro d'identification appelé ***inode*** (figure suivante). Chaque nom de fichier est donc associé à un inode (son contenu).



Fichiers associés à des inodes

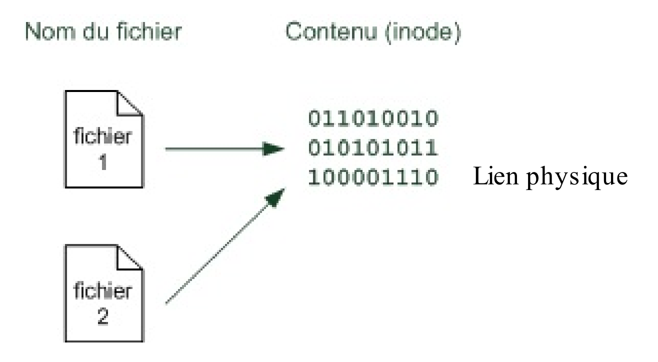
C'est tout ce que vous avez besoin de savoir pour comprendre la suite.

Nous allons maintenant découvrir comment créer des liens physiques puis des liens symboliques.

Créer des liens physiques

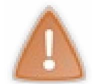
Ce type de lien est plus rarement utilisé que le lien symbolique, mais il faut tout de même le connaître car il peut se révéler pratique.

Un lien physique permet d'avoir deux noms de fichiers qui partagent exactement le même contenu, c'est-à-dire le même inode (figure suivante).



Lien physique

Ainsi, que vous passiez par `fichier1` ou par `fichier2`, vous modifiez exactement le même contenu. En quelque sorte, **le fichier est le même**. On peut juste y accéder via deux noms de fichiers différents.



On ne peut pas créer de liens physiques sur des répertoires. Cela ne fonctionne qu'avec les fichiers.
Il existe des options pour que ça fonctionne avec des répertoires, mais c'est un peu particulier et on n'en parlera pas.
Pour faire un « raccourci » vers un répertoire, on préfèrera utiliser un lien symbolique.

Pour créer un lien physique, nous allons utiliser la commande `ln`. Je vous propose tout d'abord de créer un répertoire pour nos tests :

Code : Console

```
mkdir tests
cd tests
```

Une fois dans ce dossier, créez un fichier avec la commande `touch` par exemple :

Code : Console

```
touch fichier1
```

Nous voulons maintenant créer un lien physique : nous allons créer un `fichier2` qui partagera le même inode (le même contenu) que `fichier1`. Tapez :

Code : Console

```
ln fichier1 fichier2
```

Si vous listez les fichiers du répertoire, vous avez l'impression d'avoir deux fichiers différents :

Code : Console

```
mateo21@mateo21-desktop:~/tests$ ls -l
total 0
-rw-r--r-- 2 mateo21 mateo21 0 2008-07-31 13:55 fichier1
-rw-r--r-- 2 mateo21 mateo21 0 2008-07-31 13:55 fichier2
```

A priori, rien ne nous permet ici de deviner que ces fichiers modifient le même contenu. Le lien physique est donc un lien dur, pas évident à détecter au premier coup d'œil.



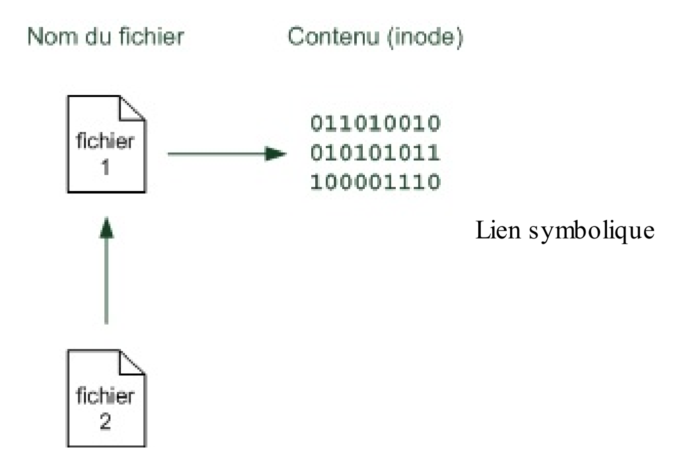
La seconde colonne de la liste (qui indique « 2 » pour chacun des fichiers) correspond au nombre de fichiers qui partagent le même inode. C'est le seul indice qui vous permet de savoir que quelqu'un a fait un lien physique, mais vous ne pouvez pas savoir lequel. Le seul moyen de vérifier que ces fichiers partagent le même contenu, c'est de faire `ls -i` pour afficher les numéros d'inode correspondants et de vérifier que ces deux fichiers sont associés au même inode.
En temps normal, sur la plupart des fichiers la seconde colonne indique donc « 1 ». Si c'est un dossier, ce nombre indique en revanche le nombre de fichiers à l'intérieur.

Si vous supprimez un des deux fichiers, l'autre fichier reste en place et le contenu sera toujours présent sur le disque. L'inode est supprimé uniquement quand plus aucun nom de fichier ne pointe dessus.
En clair, supprimez `fichier1` pour voir. Vous verrez que `fichier2` existe toujours et qu'il affiche toujours le même contenu. Il faut supprimer `fichier1` **ET** `fichier2` pour supprimer le contenu.

Créer des liens symboliques

Les liens symboliques ressemblent plus aux « raccourcis » dont vous avez peut-être l'habitude sous Windows. La plupart du temps, on crée des liens symboliques sous Linux pour faire un raccourci, et non des liens physiques qui sont un peu particuliers.

Le principe du lien symbolique est que l'on crée un lien vers un autre nom de fichier. Cette fois, on pointe vers le nom de fichier et non vers l'inode directement (figure suivante).



Lien symbolique

Supprimez le `fichier2` que nous avons créé tout à l'heure (sous forme de lien physique) :

Code : Console

```
rm fichier2
```

Créons maintenant un nouveau `fichier2`, cette fois sous forme de lien symbolique. On utilise là encore la commande `ln`, mais avec le paramètre `-s` (s comme symbolique) :

Code : Console

```
ln -s fichier1 fichier2
```

Cette fois, la commande détaillée `ls -l` sera beaucoup plus précise :

Code : Console

```
mateo21@mateo21-desktop:~/tests$ ls -l
total 0
-rw-r--r-- 1 mateo21 mateo21 0 2008-07-31 13:55 fichier1
lrwxrwxrwx 1 mateo21 mateo21 8 2008-07-31 14:15 fichier2 -> fichier1
```

On note deux choses :

- la toute première lettre de la seconde ligne est un `l` (comme *link*, c'est-à-dire lien) ;
- tout à la fin de la seconde ligne, une flèche montre clairement que `fichier2` pointe vers `fichier1`.

Bref, les liens symboliques sont beaucoup plus faciles à repérer que les liens physiques !


Ok, mais quelles différences à part ça ? Le résultat revient au même, non ? Qu'on ouvre `fichier1` ou `fichier2`, on éditera le même contenu au final !

Tout à fait. Il y a quand même quelques subtilités :

- par exemple, si vous supprimez `fichier2`, il ne se passe rien de mal. Par contre, si vous supprimez `fichier1`, `fichier2` pointera vers un fichier qui n'existe plus. Le lien symbolique sera cassé et ne servira donc plus à rien. On parle de « lien mort » ;
- d'autre part, l'avantage des liens symboliques est qu'ils fonctionnent aussi sur des répertoires, contrairement aux liens physiques.

En résumé

- `cat` permet d'afficher tout le contenu d'un fichier, mais lorsque celui-ci est long, il est préférable d'utiliser `less` qui affiche le fichier page par page.
- On peut obtenir uniquement le début ou la fin d'un fichier avec `head` et `tail`. En utilisant `tail -f` on peut suivre l'évolution d'un fichier en temps réel, ce qui est utile sur les fichiers de log qui enregistrent l'activité du système.
- `mkdir` permet de créer un dossier, `touch` permet de créer un fichier vide.
- `cp` permet de copier un fichier ou un dossier, tandis que `mv` permet de les déplacer ou de les renommer.
- `rm` supprime un fichier. Il n'y a pas de corbeille en console, la suppression est définitive ; il faut donc être prudent.
- On peut créer des liens (raccourcis) vers des fichiers et dossiers à l'aide de la commande `ln`.

 Les utilisateurs et les droits

Linux est un système multi-utilisateurs. Cela signifie que plusieurs personnes peuvent travailler simultanément sur le même OS, en s'y connectant à distance notamment.

Puisque plusieurs utilisateurs peuvent être connectés à Linux en même temps, celui-ci doit avoir une excellente organisation dès le départ. Ainsi chaque personne a **son** propre compte utilisateur, et il existe un ensemble de règles qui disent qui a le droit de faire quoi.
Je vous propose de découvrir tous ces mécanismes dans ce chapitre.

sudo: exécuter une commande en root

Lorsque vous avez installé Ubuntu, on vous a demandé le nom du compte utilisateur que vous vouliez créer. Par exemple dans mon cas j'ai créé l'utilisateur « mateo21 ».
Dans la plupart des distributions Linux on vous proposera de créer un compte utilisateur avec des **droits limités**, comme c'est le cas pour mon compte « mateo21 ».



Attends, c'est nous qui avons installé Linux mais on n'a pas le droit de faire tout ce que l'on veut dessus ?

Oui, et c'est une sécurité. Bien sûr, comme vous êtes aux commandes, vous pouvez à tout moment dire : « Bon allez on passe en mode chef-qui-peut-tout-faire ». Mais c'est une sécurité de ne pas avoir le droit de tout faire par défaut, car certaines commandes peuvent être dangereuses pour la stabilité et la sécurité de votre ordinateur. Avoir des droits limités, cela signifie aussi qu'on s'empêche par exemple d'exécuter la « commande de la mort qui tue » qu'on a vue dans le chapitre précédent (`rm -rf /*`).

Nous allons d'abord commencer par voir comment sont organisés les utilisateurs sous Linux, puis nous verrons comment devenir le « chef ».
Plus loin dans le chapitre, nous apprendrons à créer et supprimer des utilisateurs en ligne de commande.

L'organisation des utilisateurs sous Linux

On peut créer autant d'utilisateurs que l'on veut, eux-mêmes répartis dans des groupes.
Il y a un utilisateur « spécial », **root**, aussi appelé superutilisateur. Celui-ci a tous les droits sur la machine.

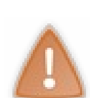
Vous pouvez voir ce que ça donne sur la figure suivante.

root
Super-utilisateur
root
Groupe
famille
amis
miguel
patrick
thomas
Utilisateur
mateo21
caroline

Au départ, chez moi, deux utilisateurs sont créés : root et mateo21.

On ne se connecte en root que très rarement, **lorsque c'est nécessaire**. Certaines commandes de Linux que nous allons voir dans ce chapitre ne sont accessibles qu'à root.
Le reste du temps, on utilise le compte « limité » que l'on a créé (mateo21 dans mon cas).

Cette simple protection permet de largement limiter les dégâts en cas de fausse manipulation, de virus sur votre PC, etc. En effet, un virus ne peut rien faire de plus que vous quand vous êtes connectés avec des droits limités. En revanche, si vous êtes en root il pourra tout faire, même détruire votre ordinateur.
Sous Windows, vous êtes toujours connectés en administrateur par défaut (équivalent de root), ce qui explique pourquoi les virus y sont si dangereux.



Exception : Ubuntu est une des rares distributions à interdire de se connecter (logger) en root. Le compte root existe mais vous n'y avez pas accès directement. Nous allons voir que ce n’est pas un problème puisqu’on peut y accéder indirectement.
Les développeurs d'Ubuntu justifient ce choix car ils considèrent, à juste titre, qu'il est dangereux de laisser le compte root entre les mains d'un débutant. Moi-même sur d'autres distributions j'ai tendance à désactiver l'accès direct à l'utilisateur root.

sudo : devenir root un instant

Par défaut, vous êtes connectés sous votre compte limité (mateo21 pour ma part).
Il est impossible sous Ubuntu de se connecter directement en root au démarrage de l'ordinateur. Comment faire alors pour exécuter des commandes que seul root a le droit d'exécuter ?

On peut devenir root **temporairement** à l'aide de la commande `sudo`.
Cette commande signifie « Faire en se substituant à l'utilisateur » : **S**ubstitute **U**ser **DO**.

Écrivez donc `sudo` suivi de la commande que vous voulez exécuter, comme ceci :

Code : Console

```
sudo commande
```

On vous demandera normalement votre mot de passe (au moins la première fois) pour exécuter la commande. Ce mot de passe est le même que celui de votre compte utilisateur limité.

Par exemple, vous pouvez exécuter un simple `ls` avec les droits root (vous ne risquez rien, rassurez-vous) :

Code : Console

```
mateo21@mateo21-desktop:/home$ sudo ls
[sudo] password for mateo21:
autredossier   Desktop    Examples  Images  Modèles      Musique  tutos
autresanimaux  Documents  images    log     mondossier   Public   Vidéos
```

Comme vous le voyez, on vous demande d'abord le mot de passe, par sécurité.
Faire un `ls` en tant que root n'apporte rien de bien spécial, c'était simplement pour avoir un exemple « sûr » avec lequel vous ne risquez pas d'endommager votre ordinateur.

`sudo su` : devenir root et le rester

Si vous tapez `sudo su` (tout court), vous passerez root indéfiniment.

Code : Console

```
mateo21@mateo21-desktop:/home$ sudo su
[sudo] password for mateo21:
root@mateo21-desktop:/home#
```

Le symbole # à la fin de l'invite de commandes vous indique que vous êtes devenus superutilisateur.
Vous pouvez alors exécuter autant de commandes en root que vous le voulez.

Pour quitter le « mode root », tapez `exit` (ou faites la combinaison `Ctrl + D`).

Code : Console

```
root@mateo21-desktop:/home/mateo21# exit
exit
mateo21@mateo21-desktop:~$
```

Et vous voilà redevenus simples mortels.



Sous les autres distributions qu'Ubuntu, écrire « `su` » suffit à passer root.
Il est néanmoins recommandé dans ce cas d'ajouter un tiret en paramètre, c'est-à-dire d'écrire « `su -` ». L'ajout du tiret a pour effet de rendre accessibles certains programmes destinés seulement à root. Par ailleurs, cela vous place directement dans le dossier personnel de root (`/root`).

adduser : gestion des utilisateurs

Maintenant que vous savez passer root (temporairement ou indéfiniment), nous allons pouvoir découvrir des commandes qui sont réservées à root.
`adduser` et `deluser` sont de celles-là. Si vous essayez de les appeler avec votre utilisateur normal, on vous dira que vous n'avez pas le droit de les utiliser. Seul root peut gérer les utilisateurs.

`adduser` : ajouter un utilisateur

La commande `adduser` permet d'ajouter un utilisateur. Vous devez au minimum fournir un paramètre : le nom de l'utilisateur à créer.
Par exemple, pour créer un compte pour Patrick :

Code : Console

```
root@mateo21-desktop:/home# adduser patrick
Ajout de l'utilisateur « patrick »...
Ajout du nouveau groupe « patrick » (1001)...
Ajout du nouvel utilisateur « patrick » (1001) avec le groupe « patrick »...
Création du répertoire personnel « /home/patrick »...
Copie des fichiers depuis « /etc/skel »...
```

Pensez à rajouter un `sudo` devant la commande si vous n'êtes pas déjà root ; pour cela, tapez `sudo adduser patrick`.
Moi je n'ai pas eu à le faire car j'ai choisi de rester root indéfiniment en tapant `sudo su` auparavant.
Si vous tentez d'exécuter la commande avec votre compte limité, vous aurez une erreur de ce genre : « `adduser` : Seul le superutilisateur peut ajouter un utilisateur ou un groupe sur le système ».

Le répertoire personnel de patrick est automatiquement créé (`/home/patrick`) et son compte est préconfiguré.
On vous demande ensuite de taper son mot de passe :

Code : Console

```
Entrez le nouveau mot de passe UNIX :
Retapez le nouveau mot de passe UNIX :
passwd : le mot de passe a été mis à jour avec succès
```

Tapez le mot de passe de patrick puis faites `Entrée`. Retapez-le pour valider.
Encore une fois, si vous ne voyez pas d'étoiles `*` quand vous tapez le mot de passe, c'est normal ; c'est une sécurité pour qu'on ne puisse pas compter le nombre de caractères derrière votre épaule.

On vous propose ensuite de rentrer quelques informations personnelles sur patrick, comme son nom, son numéro de téléphone… Si vous voulez le faire, faites-le, mais sinon sachez que vous pouvez taper `Entrée` sans rien écrire ; on ne vous embêtera pas.

Code : Console

```
Modification des informations relatives à l'utilisateur patrick
Entrez la nouvelle valeur ou « Entrée » pour conserver la valeur proposée
        Nom complet []:
        N° de bureau []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Ces informations sont-elles correctes ? [o/N] o
```

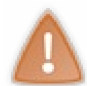
À la fin, on vous demande de confirmer par un « `o` » (oui) que tout est bon. Tapez `Entrée` et ça y est, le compte de patrick est créé !

`passwd` : changer le mot de passe

S'il était nécessaire de changer le mot de passe de patrick par la suite, utilisez la commande `passwd` en indiquant en paramètre le nom du compte à modifier.

Code : Console

```
root@mateo21-desktop:/home# passwd patrick
Entrez le nouveau mot de passe UNIX :
Retapez le nouveau mot de passe UNIX :
passwd : le mot de passe a été mis à jour avec succès
```



Attention ! Si vous appelez `passwd` sans préciser de compte en paramètre, c'est le mot de passe de l'utilisateur sous lequel vous êtes connecté que vous changerez ! Ainsi, si vous êtes en root, c'est le mot de passe de root qui sera modifié.

`deluser` : supprimer un compte

patrick vous ennuie ? patrick est parti ? Si son compte n'est plus nécessaire (ou que vous voulez vous venger) vous pouvez le supprimer avec `deluser`.

Code : Console

```
deluser patrick
```

Aucune confirmation ne vous sera demandée !



Ne supprimez en aucun cas votre compte utilisateur ! Par exemple, je ne dois surtout pas supprimer le compte mateo21. En effet, si je le fais, il n'y aura plus que root sur la machine… et Ubuntu interdit de se `logger` en root. Par conséquent, au prochain démarrage de la machine vous ne pourrez pas vous connecter… et vous serez complètement coincés !

Toutefois, cette commande seule ne supprime pas le répertoire personnel de patrick. Si vous voulez supprimer aussi son `home` et tous ses fichiers personnels, utilisez le paramètre `--remove-home` :

Code : Console

```
deluser --remove-home patrick
```



`adduser` et `deluser` sont des commandes qui n'existent que sous Debian et tous ses descendants, dont Ubuntu. Partout ailleurs on doit utiliser `useradd` et `userdel`, qui sont les commandes Unix traditionnelles fonctionnant partout. Elles font globalement la même chose mais de manière beaucoup plus basique : si vous n'appelez pas `passwd` vous-mêmes, le compte ne sera pas activé et n'aura pas de mot de passe.

addgroup : gestion des groupes

Je vous l'ai dit au début : chaque utilisateur appartient à un groupe.



Oui mais dans ce cas, à quel groupe appartiennent les utilisateurs mateo21 et patrick ? On n'a rien défini, nous !

En effet, si vous ne définissez rien, un groupe du même nom que l'utilisateur sera automatiquement créé : ainsi, mateo21 appartient au groupe mateo21 et patrick au groupe patrick.

On peut le vérifier en regardant à qui appartiennent les dossiers dans `/home` via un `ls -l` :

Code : Console

```
root@mateo21-desktop:~# cd /home
root@mateo21-desktop:/home# ls -l
total 24
drwx------  2 root    root    16384 2007-09-19 18:22 lost+found
drwxr-xr-x 65 mateo21 mateo21  4096 2007-11-15 22:40 mateo21
drwxr-xr-x  2 patrick patrick  4096 2007-11-15 23:00 patrick
```

Souvenez-vous : la 3ème colonne indique le propriétaire du fichier ou dossier ; la 4ème indique le groupe qui possède ce fichier ou dossier.

Ainsi, le dossier mateo21 appartient à l'utilisateur mateo21 et au groupe mateo21.
Même chose pour patrick.
On constatera par ailleurs que `lost+found` appartient à root et qu'il y a un groupe root (root fait donc partie du groupe root).

Bon, mais quel intérêt y a-t-il à ce que tout le monde soit dans son propre groupe, me direz-vous ?
Vous pourriez très bien vous contenter de ce fonctionnement (un utilisateur = un groupe), mais au cas où vous auriez beaucoup d'utilisateurs, je vais quand même vous montrer comment créer des groupes.

addgroup : créer un groupe

La commande `addgroup` crée un nouveau groupe. Vous avez juste besoin de spécifier le nom de celui-ci en paramètre :

Code : Console

```
root@mateo21-desktop:/home# addgroup amis
Ajout du groupe « amis » (identifiant 1002)...
Terminé.
```

Super, mais personne ne fait encore partie de ce groupe. :-(

usermod : modifier un utilisateur

La commande `usermod` permet d'éditer un utilisateur. Elle possède plusieurs paramètres ; nous allons en retenir deux :

- **-l** : renomme l'utilisateur (le nom de son répertoire personnel ne sera pas changé par contre) ;
- **-g** : change de groupe.

Si je veux mettre patrick dans le groupe `amis`, je ferai donc comme ceci :

Code : Console

```
usermod -g amis patrick
```

Et pour remettre patrick dans le groupe patrick comme il l'était avant :

Code : Console

```
usermod -g patrick patrick
```



Il est aussi possible de faire en sorte qu'un utilisateur appartienne à plusieurs groupes. Pour ce faire, utilisez le paramètre `-G` (majuscule).



Exemple : `usermod -G amis,paris,collegues patrick`.
Séparez les noms des groupes par une virgule, sans espace entre chaque nom de groupe.



Faites très attention en utilisant `usermod` ! Lorsque vous avez recours à `-G`, l'utilisateur change de groupe et ce peu importe les groupes auxquels il appartenait auparavant.
Si vous voulez **ajouter** des groupes à un utilisateur (sans perdre les groupes auxquels il appartenait avant cela), utilisez `-a` :
`usermod -aG amis patrick`

`delgroup` : supprimer un groupe

Si vous voulez supprimer un groupe, c'est tout simple :

Code : Console

```
delgroup amis
```



`addgroup` et `delgroup` n'existent que sous Debian et ses dérivés (même remarque que pour `adduser` et `deluser`).
Les commandes « traditionnelles » qui fonctionnent partout sont `groupadd` et `groupdel`, mais elles offrent moins d'options.

chown : : gestion des propriétaires d'un fichier

Seul l'utilisateur root peut changer le propriétaire d'un fichier.
Supposons par exemple que mateo21 possède dans son répertoire personnel un fichier appelé `rapport.txt`.

Voici le résultat d'un `ls -l` pour ce fichier :

Code : Console

```
mateo21@mateo21-desktop:~$ ls -l rapport.txt
-rw-r--r-- 1 mateo21 mateo21 0 2007-11-15 23:14 rapport.txt
```



Petite astuce : comme vous venez de le voir, si on précise un nom de fichier en dernier paramètre de la commande `ls`, on ne verra que ce fichier dans les résultats.
Le joker `*` est là aussi utilisable : `ls -l *.jpg` afficherait uniquement les images JPEG contenues dans ce dossier.

Ce fichier, je souhaite le « donner » à patrick. C'est là qu'intervient la commande `chown`.

`chown` : changer le propriétaire d'un fichier

La commande `chown`, qui doit être utilisée **en tant que root**, attend deux paramètres au moins :

- le nom du nouveau propriétaire ;
- le nom du fichier à modifier.

Cela donne donc :

Code : Console

```
chown patrick rapport.txt
```

On peut voir ensuite que patrick est bien le nouveau propriétaire du fichier :

Code : Console

```
root@mateo21-desktop:/home/mateo21# ls -l rapport.txt
-rw-r--r-- 1 patrick mateo21 0 2007-11-15 23:14 rapport.txt
```

Seulement… il appartient toujours au groupe mateo21 !

`chgrp` : changer le groupe propriétaire d'un fichier

`chgrp` s'utilise exactement de la même manière que `chown` à la différence près qu'il affecte cette fois le groupe propriétaire d'un fichier.

Code : Console

```
chgrp amis rapport.txt
```

Cette commande affectera le fichier `rapport.txt` au groupe `amis`.

Un petit `ls -l` nous confirmera que `rapport.txt` appartient désormais à patrick et au groupe `amis` :

Code : Console

```
root@mateo21-desktop:/home/mateo21# ls -l rapport.txt
-rw-r--r-- 1 patrick amis 0 2007-11-15 23:14 rapport.txt
```

`chown` peut aussi changer le groupe propriétaire d'un fichier !

Eh oui ! C'est d'ailleurs l'astuce que j'utilise le plus souvent :

Code : Console

```
chown patrick:amis rapport.txt
```

Cela affectera le fichier à l'utilisateur patrick et au groupe `amis`.
Il suffit de séparer par un symbole deux points (« : ») le nom du nouvel utilisateur (à gauche) et le nom du nouveau groupe (à droite).

`-R` : affecter récursivement les sous-dossiers

Très utile aussi, l'option `-R` de `chown`. Elle modifie tous les sous-dossiers et fichiers contenus dans un dossier pour y affecter un nouvel utilisateur (et un nouveau groupe si on utilise la technique du deux points que l'on vient de voir).

Par exemple, si je suis sadique et que je veux donner tout le contenu du dossier personnel de patrick à mateo21 (et au groupe mateo21), c'est très simple :

Code : Console

```
chown -R mateo21:mateo21 /home/patrick/
```

Résultat :

Code : Console

```
root@mateo21-desktop:/home# ls -l
total 24
drwx------  2 root    root    16384 2007-09-19 18:22 lost+found
drwxr-xr-x 62 mateo21 mateo21  4096 2007-11-15 23:19 mateo21
drwxr-xr-x  2 mateo21 mateo21  4096 2007-11-15 23:00 patrick
```

Désormais tous les fichiers à l'intérieur du dossier de patrick appartiennent à mateo21 (je sais, je suis vraiment trop diabolique).

chmod : modifier les droits d'accès

On attaque maintenant la partie la plus « coton » du chapitre si je puis dire : les droits d'accès.

Le fonctionnement des droits

Chaque fichier et chaque dossier possède une liste de droits. C'est une liste qui indique qui a le droit de voir le fichier, de le modifier et de l'exécuter.

Vous avez déjà vu des listes de droits, oui oui ! Lorsque vous faites un `ls -l`, il s'agit de la première colonne :

Code : Console

```
mateo21@mateo21-desktop:~$ ls -l
total 40
drwxr-xr-x 2 mateo21 mateo21 4096 2007-11-13 21:53 Desktop
drwxr-xr-x 2 mateo21 mateo21 4096 2007-11-13 13:46 Documents
lrwxrwxrwx 1 mateo21 mateo21   26 2007-09-19 18:31 Examples -
> /usr/share/example-content
drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-25 20:28 images
drwxr-xr-x 2 mateo21 mateo21 4096 2007-10-19 01:21 Images
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-25 11:11 log
drwxr-xr-x 2 mateo21 mateo21 4096 2007-10-19 01:21 Modèles
drwxr-xr-x 2 mateo21 mateo21 4096 2007-10-19 01:21 Musique
drwxr-xr-x 2 mateo21 mateo21 4096 2007-10-19 01:21 Public
-rw-r--r-- 1 mateo21 mateo21    0 2007-11-15 23:14 rapport.txt
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-19 19:51 tutos
drwxr-xr-x 2 mateo21 mateo21 4096 2007-10-19 01:21 Vidéos
```

Vous voyez tous ces `d`, `r`, `w` et `x` au début ? Ce sont ce qu'on appelle les droits d'accès du fichier ou dossier.

On peut voir cinq lettres différentes. Voici leur signification :

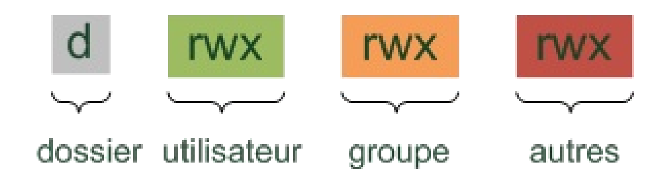
- **d** (Directory) : indique si l'élément est un dossier ;
- **l** (Link) : indique si l'élément est un lien (raccourci) ;
- **r** (Read) : indique si on peut lire l'élément ;
- **w** (Write) : indique si on peut modifier l'élément ;
- **x** (eXecute) : si c'est un fichier, « `x` » indique qu'on peut l'exécuter. Ce n'est utile que pour les fichiers exécutables (programmes et scripts).
  Si c'est un dossier, « `x` » indique qu'on peut le « traverser », c'est-à-dire qu'on peut voir les sous-dossiers qu'il contient si on a le droit de lecture dessus.

Si la lettre apparaît, c'est que le droit existe. S'il y a un tiret à la place, c'est qu'il n'y a aucun droit.



Pourquoi est-ce qu'on voit parfois `r`, `w` et `x` à plusieurs reprises ?

Les droits sont découpés en fonction des utilisateurs (figure suivante).



Le premier élément `d` mis à part, on constate que `r`, `w` et `x` sont répétés trois fois en fonction des utilisateurs :

- le premier triplet `rwx` indique les droits que possède le **propriétaire** du fichier sur ce dernier ;
- le second triplet `rwx` indique les droits que possèdent les autres membres du **groupe** sur ce fichier ;
- enfin, le dernier triplet `rwx` indique les droits que possèdent tous les **autres** utilisateurs de la machine sur le fichier.

Prenons un cas concret, le fichier `rapport.txt` :

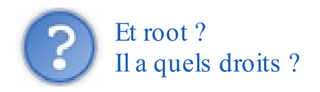
Code : Console

```
mateo21@mateo21-desktop:~$ ls -l rapport.txt
-rw-r--r-- 1 mateo21 mateo21 0 2007-11-15 23:14 rapport.txt
```

Ses droits sont : `-rw-r--r--`

- **-** : le premier tiret indique qu'il ne s'agit pas d'un dossier. S'il y avait eu un `d` à la place, cela aurait indiqué qu'il s'agissait d'un dossier.
- **rw-** : indique que le propriétaire du fichier, mateo21 en l'occurrence, peut lire et modifier (et donc supprimer) le fichier. En revanche, il ne peut pas l'exécuter car il n'a pas de `x` à la fin. Je rappelle que quiconque peut modifier un fichier a aussi le droit de le supprimer.
- **r--** : tous les utilisateurs qui font partie du groupe mateo21 mais qui ne sont pas mateo21 peuvent seulement lire le fichier. Ils ne peuvent ni le modifier, ni l'exécuter. Je reconnais qu'avoir un nom de groupe identique au nom d'utilisateur peut embrouiller : si vous êtes aussi bien organisés que sur mon premier schéma, on parlera plutôt du groupe famille.
- **r--** : tous les autres (ceux qui ne font pas partie du groupe mateo21) peuvent seulement lire le fichier.

En résumé, ces droits nous apprennent que l'élément est un fichier, que mateo21 peut le lire et le modifier et que tous les autres utilisateurs peuvent seulement le lire.



Et root ?
Il a quels droits ?

Souvenez-vous d'une chose : root a TOUS les droits. Il peut tout faire : lire, modifier, exécuter n'importe quel fichier.

`chmod` : modifier les droits d'accès

Maintenant que nous savons voir et comprendre les droits d'accès d'un fichier, nous allons apprendre à les modifier à l'aide de la commande `chmod`.

Une précision importante pour commencer : contrairement aux commandes précédentes, vous n'avez pas besoin d'être root pour utiliser `chmod`. Vous devez juste être propriétaires du fichier dont vous voulez modifier les droits d'accès.

`chmod` est un petit peu délicat à utiliser. En effet, on peut attribuer les droits sur un fichier / dossier via plusieurs méthodes différentes, la plus courante étant celle des chiffres.

Attribuer des droits avec des chiffres (`chmod` absolu)

J'espère que vous êtes prêts pour effectuer quelques additions !
Il va falloir faire un petit peu de calcul mental. En effet, on attribue un chiffre à chaque droit :

Droit	Chiffre
r	4
w	2
x	1

Si vous voulez combiner ces droits, il va falloir additionner les chiffres correspondants.
Ainsi, pour attribuer le droit de lecture et de modification, il faut additionner $4+2$, ce qui donne 6. Le chiffre 6 signifie donc « Droit de lecture et d'écriture ».

Voici la liste des droits possibles et la valeur correspondante :

Droits	Chiffre	Calcul
---	0	0 + 0 + 0
r--	4	4 + 0 + 0
-w-	2	0 + 2 + 0
--x	1	0 + 0 + 1
rw-	6	4 + 2 + 0
-wx	3	0 + 2 + 1
r-x	5	4 + 0 + 1
rwx	7	4 + 2 + 1

C'est compris ?
Avec ça, on peut calculer la valeur d'un triplet de droits. Il faut faire le même calcul pour les droits que l'on veut attribuer au propriétaire, au groupe et aux autres.

Par exemple, « 640 » indique les droits du propriétaire, du groupe et des autres (dans l'ordre).

- 6 : droit de lecture et d'écriture pour le propriétaire.
- 4 : droit de lecture pour le groupe.
- 0 : aucun droit pour les autres.

Le droit maximal que l'on puisse donner à tout le monde est 777 : droit de lecture, d'écriture et d'exécution pour le propriétaire, pour son groupe et pour tous les autres. Bref, avec un tel droit tout le monde peut tout faire sur ce fichier.
Au contraire, avec un droit de 000, personne ne peut rien faire… à part root, bien sûr.

Pour changer les droits sur le fichier `rapport.txt`, et être le seul autorisé à le lire et l'éditer, je dois exécuter cette commande :

Code : Console

```
chmod 600 rapport.txt
```

Un petit `ls -l` pour voir le résultat :

Code : Console

```
mateo21@mateo21-desktop:~$ ls -l rapport.txt
-rw------- 1 mateo21 mateo21 0 2007-11-15 23:14 rapport.txt
```

Bingo !
On a bien confirmation que seul le propriétaire du fichier, c'est-à-dire moi, peut le lire et le modifier !

Attribuer des droits avec des lettres (`chmod` relatif)

Il existe un autre moyen de modifier les droits d'un fichier. Il revient un peu au même mais permet parfois de paramétrer plus finement, droit par droit.
Dans ce mode, il faut savoir que :

- **u** = user (propriétaire) ;
- **g** = group (groupe) ;
- **o** = other (autres).

… et que :

- **+** signifie : « Ajouter le droit » ;
- **-** signifie : « Supprimer le droit » ;
- **=** signifie : « Affecter le droit ».

Maintenant que vous savez cela, vous pouvez écrire :

Code : Console

```
chmod g+w rapport.txt
```

Signification : « Ajouter le droit d'écriture au groupe ».

Code : Console

```
chmod o-r rapport.txt
```

Signification : « Enlever le droit de lecture aux autres ».

Code : Console

```
chmod u+rx rapport.txt
```

Signification : « Ajouter les droits de lecture et d'exécution au propriétaire ».

Code : Console

```
chmod g+w,o-w rapport.txt
```

Signification : « Ajouter le droit d'écriture au groupe et l'enlever aux autres ».

Code : Console

```
chmod go-r rapport.txt
```

Signification : « Enlever le droit de lecture au groupe et aux autres ».

Code : Console

```
chmod +x rapport.txt
```

Signification : « Ajouter le droit d'exécution à tout le monde ».

Code : Console

```
chmod u=rwx,g=r,o=- rapport.txt
```

Signification : « Affecter tous les droits au propriétaire, juste la lecture au groupe, rien aux autres ».

Voilà, ouf ! J'ai préféré vous expliquer le fonctionnement à travers des exemples concrets plutôt que de faire un cours théorique sur la syntaxe d'une des utilisations possibles de `chmod`.
Normalement si vous suivez mes exemples vous devriez être capables de tout faire !

Et toujours… `-R` pour affecter récursivement

Le paramètre `-R` existe aussi pour `chmod`. Si vous affectez des droits sur un dossier avec `-R`, tous ses fichiers et sous-dossiers récupèreront le même droit.

Si je veux être le seul à pouvoir lire, éditer et exécuter les fichiers de mon répertoire personnel et de tous ses fichiers, j'ai juste besoin d'écrire :

Code : Console

```
chmod -R 700 /home/mateo21
```

C'est tout !

En résumé

- Chaque personne qui utilise une machine Linux possède un compte utilisateur.
- Les utilisateurs sont classés par groupes.
- Il existe un superutilisateur qui a tous les droits : root. C'est l'administrateur de la machine, le seul à être autorisé à installer des programmes ou effectuer certaines modifications sur le système.
- Certaines commandes ne fonctionnent que lorsqu'on est root et nécessitent donc de se transformer en root à l'aide de `sudo`. C'est le cas de la commande d'ajout d'utilisateur (`adduser`), de suppression d'utilisateur (`deluser`) ou encore de changement de propriétaire d'un fichier (`chown`).
- On peut modifier les droits d'accès à un fichier avec `chmod`. Il existe trois types de droits : `r` (droit de lecture), `w` (droit d'écriture) et `x` (droit d'exécution).

 Nano, l'éditeur de texte du débutant

Nousavons découvert plusieurs façons de voir le contenu d'un fichier en console.
Mais… aucune des commandes que nous avons étudiées ne nous permettait d'éditer un fichier.

Pourquoi ai-je repoussé le moment où je vous parlerais des éditeurs de texte ? Parce que c'est un des domaines les plus riches de la console ! Parmi les plus célèbres éditeurs de texte console de Linux, il faut connaître : Nano, Vim et Emacs.

Que de jolis noms, n'est-ce pas ?
Des trois que je viens de citer, **Nano** est de loin le plus simple à utiliser. Ce n'est pas pour rien que ce chapitre s'intitule « Nano, l'éditeur de texte du débutant ».

Nous découvrirons Vim plus loin dans ce livre, car il est plus complexe et nécessite déjà un bon niveau.

Premiers pas avec Nano

En sciences, le terme « nano » représente une toute petite unité. Par exemple, un atome a une taille d'environ 0,1 nanomètre.

Si l'éditeur de texte que je vais vous présenter s'appelle Nano, c'est parce qu'il est tout petit. Il s'agit d'un programme très simple comparé à Vim et Emacs et il nous conviendra tout à fait pour démarrer. Il possède assez peu de fonctions par rapport aux deux autres logiciels (qui peuvent devenir de véritables machines de guerre) mais suffisamment pour commencer à vous débrouiller avec un éditeur de texte.

Nano est un éditeur de texte, pas un traitement de texte !

Savez-vous vraiment ce qu'est un éditeur de texte ? Ne le confondez-vous pas avec un traitement de texte ?

Un **éditeur de texte** est un programme qui permet de modifier des fichiers de texte brut, sans mise en forme (gras, italique, souligné…). Sous Windows, on dispose d'un éditeur de texte très basique : le Bloc-Notes. Sous Linux, on a le choix entre Nano, Vim, Emacs et bien d'autres, sachant qu'au moins un de ceux-là est installé par défaut sur la plupart des distributions.

Un **traitement de texte** est fait pour rédiger des documents mis en forme. Sous Windows, Word est le plus célèbre traitement de texte ; sous Linux, on possède l'équivalent : Open Office Writer. Ces programmes ne peuvent être utilisés qu'en mode graphique, la console ne permettant pas vraiment de faire de la mise en forme.

 Quand a-t-on besoin d'un éditeur de texte ?

Chaque fois que vous devez éditer un fichier de texte brut. Sous Windows, vous avez l'habitude de voir des fichiers de texte brut au format `.txt`. Sous Linux, vous savez que l'extension importe peu (on peut trouver des fichiers en texte brut sans extension).

Les éditeurs de texte sont parfaits pour les programmeurs en particulier : ils permettent d'éditer des fichiers `.c`, `.cpp`, `.h`, `.rb`, `.py`, etc. (En fonction de votre langage de programmation.)
Même si vous ne programmez pas, vous aurez besoin d'utiliser un éditeur de texte pour modifier des fichiers de configuration. Ces fichiers n'ont pas d'extension particulière, mais à force vous apprendrez à les reconnaître.

Après avoir appris à utiliser Nano, nous nous ferons les dents sur nos premiers fichiers de configuration : le `nanorc` et le `bashrc`. Ce sera l'occasion pour vous de personnaliser votre Nano et votre console.;-)

Découverte de Nano

Le nom complet de Nano est « GNU nano », en référence au projet GNU dont je vous ai parlé dans le tout premier chapitre. Il s'agit d'un logiciel qui s'inspire de « pico », un éditeur de texte plus ancien qui se voulait lui aussi très simple d'utilisation.

Pour démarrer le logiciel, il vous suffit simplement de taper `nano` dans la console :

Code : Console

```
nano
```

L'éditeur Nano s'ouvre immédiatement (figure suivante).

```
  GNU nano 2.0.6                Nouvel espace

^G Aide      ^O Écrire    ^R Lire fich.^Y Page préc.^K Couper    ^C Pos. cur.
^X Quitter   ^J Justifier ^W Chercher  ^V Page suiv.^U Coller    ^T Orthograp.
```

Nano

Dès lors, vous pouvez commencer à taper du texte (exemple sur la figure suivante).



Nano : écriture de texte

C'est aussi simple que cela !

Ne riez pas, je précise qu'il « suffit de taper du texte » car ce n'est pas aussi simple sous d'autres éditeurs, comme Vim par exemple.

Les raccourcis clavier de Nano

En bas de votre écran, vous pouvez voir un espace d'aide (figure suivante). Que signifie-t-il exactement ?
Il s'agit d'un aide-mémoire pour vous rappeler à tout moment les commandes principales que vous pouvez lancer sous Nano.

```
^G Aide      ^O Ecrire     ^R Lire fich.^Y Page préc.^K Couper     ^C Pos. cur.
^X Quitter   ^J Justifier  ^W Chercher  ^V Page suiv.^U Coller     ^T Orthograp.
```

Aide de Nano

Le symbole ^ signifie `Ctrl` (la touche `Contrôle` de votre clavier). Ainsi, pour quitter Nano, il suffit de taper `Ctrl + X`.

Voici les raccourcis les plus importants :

- `Ctrl + G` : afficher l'aide ;
- `Ctrl + K` : couper la ligne de texte (et la mettre dans le presse-papier) ;
- `Ctrl + U` : coller la ligne de texte que vous venez de couper ;
- `Ctrl + C` : afficher à quel endroit du fichier votre curseur est positionné (numéro de ligne…) ;
- `Ctrl + W` : rechercher dans le fichier ;
- `Ctrl + O` : enregistrer le fichier (écrire) ;
- `Ctrl + X` : quitter Nano.

Vous pouvez vous déplacer dans le fichier avec les flèches du clavier ainsi qu'avec les touches `Page Up` et `Page Down` pour avancer de page en page (les raccourcis `Ctrl + Y` et `Ctrl + V` fonctionnent aussi).

Si l'aide-mémoire vous encombre, vous pouvez gagner de la place en appuyant sur `Échap` puis sur `X`. Vous pouvez l'afficher de nouveau avec la même suite de touches.

La recherche

La combinaison de touches `Ctrl + W` lance une recherche dans le fichier (figure suivante).

```
  GNU nano 2.0.6              Nouvel espace                        Modifié

Salut les Zéros, ça va ?

Je teste nano ! Un deux... un deux...

Recherche:
^G Aide        ^Y Prem. lig.^R Remplacer  ^W Début paraM-C Resp.cassM-R Exp. ratio.
^C Annuler     ^V Dern. Lig.^T Aller lig.^O Fin para.  M-B ->ArrièreP Précédente
```

Recherche dans Nano

Il vous suffit d'écrire le mot que vous recherchez (figure suivante)…

```
 GNU nano 2.0.6               Nouvel espace                        Modifié

Salut les Zéros, ça va ?

Je teste nano ! Un deux... un deux...

Recherche: deux
^G Aide       ^Y Prem. lig.^R Remplacer ^W Début paraM-C Resp.cassM-R Exp. ratio.
^C Annuler    ^V Dern. Lig.^T Aller lig.^O Fin para. M-B ->Arrière^P Précédente
```

Recherche dans Nano

… puis de taper `Entrée` (figure suivante).

```
 GNU nano 2.0.6               Nouvel espace                        Modifié

Salut les Zéros, ça va ?

Je teste nano ! Un deux... un deux...

                        [ La recherche a fait le tour ]
^G Aide       ^O Écrire     ^R Lire fich.^Y Page préc.^K Couper     ^C Pos. cur.
^X Quitter    ^J Justifier  ^W Chercher  ^V Page suiv.^U Coller     ^T Orthograp.
```

Recherche dans Nano

Le curseur est automatiquement positionné à la première occurrence trouvée. Si le curseur est à la fin, la recherche recommence du début.



Si vous voulez sortir du mode recherche, tapez `Ctrl + C` (Annuler).

Si vous souhaitez aller au résultat suivant (au « deux » suivant), faites à nouveau `Ctrl + W` pour lancer une recherche. La recherche précédente est sauvegardée et apparaît entre crochets. Si vous voulez rechercher le même mot (et donc aller au résultat suivant), tapez juste `Entrée` sans écrire de mot à rechercher (figure suivante).

```
  GNU nano 2.0.6              Nouvel espace                        Modifié

Salut les Zéros, ça va ?

Je teste nano ! Un deux... un deux...

Recherche [deux]:
^G Aide       ^Y Prem. lig.^R Remplacer ^W Début paraM-C Resp.cassM-R Exp. ratio.
^C Annuler    ^V Dern. Lig.^T Aller lig.^O Fin para. M-B ->ArrièreP^P Précédente
```

Recherche dans Nano

Enregistrer et quitter

Pour enregistrer à tout moment, faites `Ctrl + O`.
Si vous essayez de quitter (`Ctrl + X`) sans enregistrer auparavant, un message vous demandera si vous voulez sauvegarder (figure suivante).

```
  GNU nano 2.0.6              Nouvel espace                       Modifié

Salut les Zéros, ça va ?

Je teste nano ! Un deux... un deux...

Sauver l'espace modifié (RÉPONDRE « Non » EFFACERA LES CHANGEMENTS) ?
 O Oui
 N Non              ^C Annuler
```

Confirmation de sortie de Nano

Si vous appuyez sur la touche `o`, vous passerez en mode enregistrement.
Si vous appuyez sur la touche `n`, Nano quittera sans enregistrer.
Si vous utilisez la combinaison `Ctrl + C`, vous annulerez votre demande de sortie de Nano et ne quitterez donc pas le logiciel.

En appuyant sur `o`, vous vous retrouvez en mode enregistrement. Tapez juste le nom du fichier que vous voulez créer puis pressez `Entrée` (figure suivante).

```
  GNU nano 2.0.6              Nouvel espace                       Modifié

Salut les Zéros, ça va ?

Je teste nano ! Un deux... un deux...

Nom du fichier à écrire: salut.txt
^G Aide             ^T Parcourir       M-M Format Mac       M-P Ajout (au début)
^C Annuler          M-D Format DOS     M-A Ajout (à la fin) M-B Copie de sécu.
```

Enregistrement dans Nano

Après ça, Nano sera fermé et vous retrouverez votre bonne vieille ligne de commandes.

Les paramètres de la commande Nano

Lorsque vous appelez Nano dans la ligne de commandes, vous pouvez spécifier plusieurs paramètres. Le plus courant est d'indiquer en paramètre le nom du fichier qu'on veut ouvrir. Ainsi :

Code : Console

```
nano salut.txt
```

… ouvrira le fichier `salut.txt` que l'on vient de créer.

Si le fichier n'existe pas, il sera automatiquement créé par Nano lors du premier enregistrement.

À part ça, la commande `nano` accepte de nombreux paramètres. Pour vous, j'en ai sélectionné trois qui me semblent faire partie des plus utiles.

- **-m** : autorise l'utilisation de la souris sous Nano. En console, oui, oui. Vous pouvez vous en servir pour cliquer avec votre souris sur la zone de texte où vous voulez placer votre curseur.
- **-i** : indentation automatique. L'alinéa (tabulations) de la ligne précédente sera respecté lorsque vous irez à la ligne. Très utile lorsque vous éditez un fichier de code source.
- **-A** : active le retour intelligent au début de la ligne. Normalement, lorsque vous appuyez sur la touche `Origine` (aussi connue sous le nom de `Home`) située à côté de la touche `Fin`, le curseur se repositionne au tout début de la ligne. Avec cette commande, il se positionnera après les alinéas. Comme `-i`, il s'agit d'une option utile avant tout pour les programmeurs.

Si je veux lancer Nano avec toutes ces options à la fois, je peux donc écrire :

Code : Console

```
nano -miA salut.txt
```

Configurer Nano avec .nanorc

Vous savez maintenant utiliser Nano. Comme vous avez pu le voir, ce n'est pas très compliqué. Il suffit d'apprendre un peu les raccourcis clavier les plus utiles et on peut rapidement s'en servir.

Justement… et si on utilisait Nano pour quelque chose d'utile ? Non parce que bon, le fichier `salut.txt` est sympa, mais ça ne va pas nous faire avancer.

Alors pour l'occasion, je me suis dit que j'allais vous faire éditer quelques fichiers de configuration. Par exemple, il existe un fichier de configuration de Nano qui indique toutes vos préférences. Celui-ci s'appelle `.nanorc`.

Pourquoi `.nanorc` ?

La plupart des fichiers de configuration commencent par un point. Cela permet de « cacher » le fichier quand on fait un `ls`. Bien entendu, comme vous devriez maintenant le savoir, les fichiers cachés peuvent toujours être affichés en utilisant le paramètre `–a` : `ls -a`.

Chaque utilisateur de la machine peut créer son propre fichier de configuration `.nanorc` dans son répertoire personnel (home). Chez moi, ce fichier doit être placé à la position : `/home/mateo21/.nanorc`. Ce fichier est lu par Nano à chaque fois que vous le démarrez.


Je viens de regarder la liste des fichiers de mon home, mais même en incluant les fichiers cachés avec `–a` je ne vois pas de fichier appelé `.nanorc` !

En effet, il se peut que le fichier `.nanorc` n'existe pas chez vous. Si tel est le cas, Nano sera chargé avec les options par défaut.

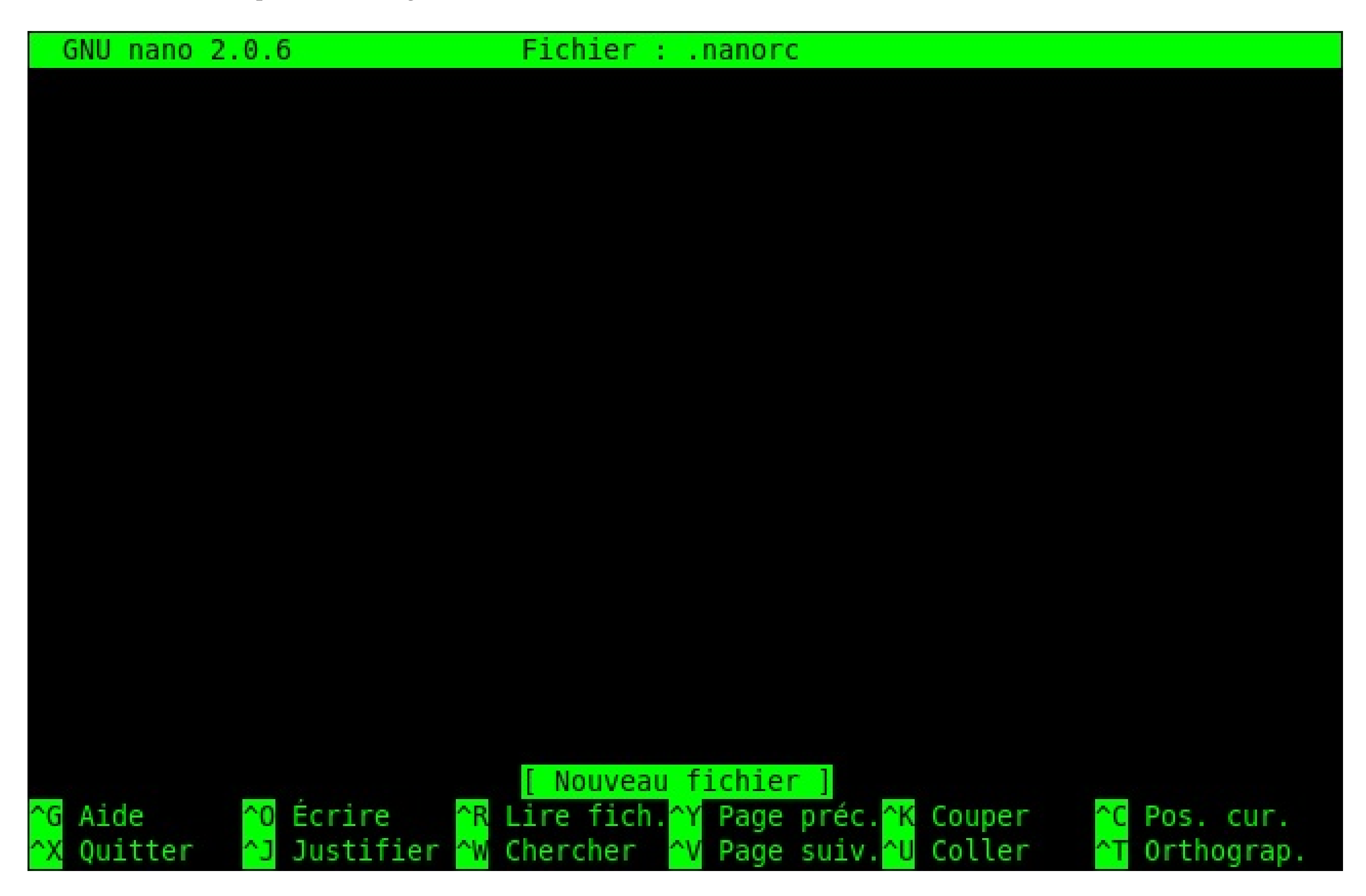
Création du .nanorc

Pas de `.nanorc` ? Pas de problème, il suffit de le créer. On peut par exemple faire ceci :

Code : Console

```
nano .nanorc
```

Cette commande ouvre Nano. Comme le fichier `.nanorc` n'existe pas, un document vide est ouvert (figure suivante). Le fichier `.nanorc` sera créé lorsque vous enregistrerez.



Dans ce fichier, vous devez écrire une commande par ligne.
Chaque commande commence par un **set** (pour activer) ou un **unset** (pour désactiver) suivi de l'option qui vous intéresse.

Par exemple, pour activer la souris, écrivez :

Code : Console

```
set mouse
```

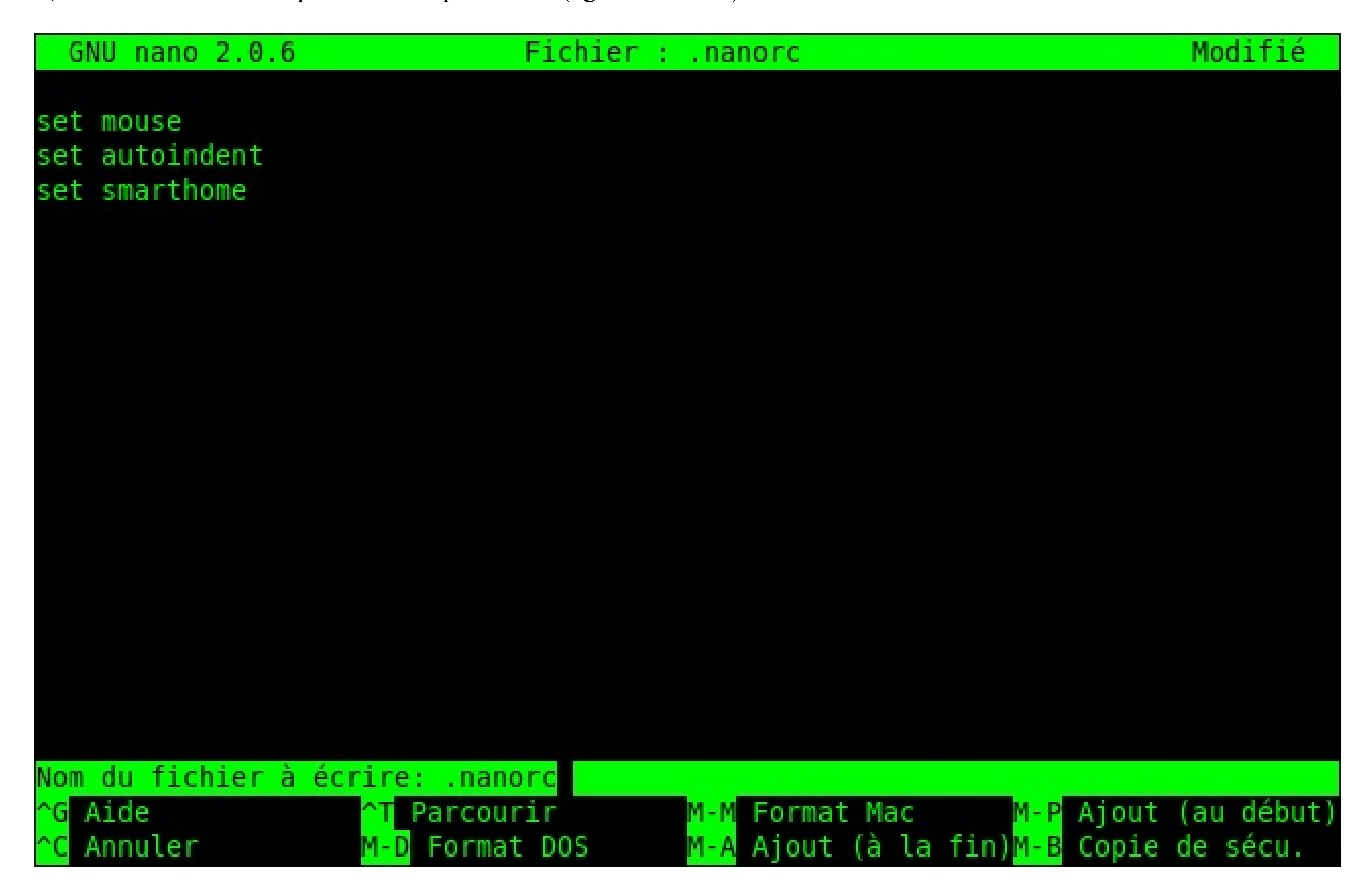
Ainsi Nano sera automatiquement chargé avec la prise en charge de la souris. Vous n'aurez pas besoin de réécrire systématiquement le paramètre `-m` qu'on a vu tout à l'heure.

On peut faire de même pour éviter d'avoir à taper à chaque fois les paramètres `-i` et `-A` avec d'autres séries de `set`. Au final, on écrira ceci :

Code : Console

```
set mouse
set autoindent
set smarthome
```

Enregistrez le fichier avec `Ctrl + O`. Comme vous avez déjà mentionné le nom du fichier en paramètre lors de l'ouverture de Nano, celui-ci sera automatiquement écrit pour vous (figure suivante).



Vous pouvez ensuite faire `Ctrl + X` pour quitter Nano.
Je vous rappelle que pour que ces options soient prises en compte, il faut démarrer une nouvelle session de Nano (c'est pour ça que la souris n'a pas automatiquement fonctionné dès que vous avez enregistré le fichier).

Si vous relancez Nano ensuite, vous pouvez constater que la souris fonctionne et que les options d'indentation automatique et de retour à la ligne intelligent sont elles aussi opérationnelles. :-)

Le `nanorc` global et la coloration syntaxique

Ce fichier `.nanorc` dans votre home est très pratique car il vous permet de définir vos propres options. Mais si vous avez dix utilisateurs sur votre machine et que vous voulez activer le support de la souris pour tout le monde, vous n'allez quand même pas créer un fichier `.nanorc` pour chacun !

Il existe un fichier `nanorc` « global » qui est pris en compte pour tout le monde. Celui-ci est situé dans `/etc/nanorc` (attention : il n'y a pas de point devant, cette fois.)
Ce fichier ne peut être modifié que par root. Je vous conseille donc de l'ouvrir avec un `sudo` (ou dans une console en root si vous avez fait `sudo su` avant) :

Code : Console

```
sudo nano /etc/nanorc
```

Normalement, ce fichier existe déjà. Comme vous pouvez le constater sur la figure suivante, il est bien rempli.

```
 GNU nano 2.0.6              Fichier : /etc/nanorc

## Sample initialization file for GNU nano.
##
## Please note that you must have configured nano with --enable-nanorc
## for this file to be read!  Also note that this file should not be in
## DOS or Mac format, and that characters specially interpreted by the
## shell should not be escaped here.
##
## To make sure a value is disabled, use "unset <option>".
##
## For the options that take parameters, the default value is given.
## Other options are unset by default.
##
## Quotes inside string parameters don't have to be escaped with
## backslashes.  The last double quote in the string will be treated as
## its end.  For example, for the "brackets" option, ""')>]}" will match
## ", ', ), >, ], and }.

## Use auto-indentation.
# set autoindent
                           [ Lecture de 263 lignes ]
^G Aide      ^O Écrire    ^R Lire fich.^Y Page préc.^K Couper     ^C Pos. cur.
^X Quitter   ^J Justifier ^W Chercher  ^V Page suiv.^U Coller     ^T Orthograp.
```

Il sert en fait de fichier d'exemple. Toutes les options disponibles dans un `.nanorc` sont présentes, mais elles sont précédées d'un # qui signifie qu'il s'agit d'un commentaire. Les commentaires sont ignorés par Nano.

Le début du fichier vous explique (en anglais) que c'est un fichier d'initialisation d'exemple de Nano.

Après le petit blabla d'introduction, vous avez la liste des options disponibles. Toutes sont commentées.
La première est `autoindent`.

Code : Console

```
# set autoindent
```

Supprimez juste le # pour décommenter la ligne et donc pour activer l'indentation automatique pour tous les utilisateurs.

Code : Console

```
set autoindent
```

Vous pouvez parcourir le fichier à la recherche d'options intéressantes que vous voulez activer.

Vers la fin, vous verrez une section appelée « color setup », qui commence par ces lignes-là :

Code : Console

```
## Nanorc files
# include "/usr/share/nano/nanorc.nanorc"
```

```
## C/C++
# include "/usr/share/nano/c.nanorc"

## HTML
# include "/usr/share/nano/html.nanorc"
```

Je vous invite à décommenter toutes les lignes d'`include`. Cela permettra d'activer la coloration « intelligente » de vos fichiers selon leur type. Vous pourrez ainsi avoir des fichiers HTML colorés, des fichiers C colorés, des fichiers `nanorc` colorés, etc.

Enregistrez le fichier puis quittez Nano.



Si vous avez une erreur lors de l'enregistrement, cela signifie que vous n'avez pas ouvert le fichier en root. Seul root a le droit de modifier ce fichier. Fermez Nano et relancez-le avec un `sudo` cette fois.

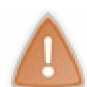
Configurer sa console avec .bashrc

Tout comme il existe un fichier de configuration de Nano, il existe un fichier de configuration de l'ensemble de la console : le `.bashrc`. Il se situe dans votre répertoire personnel et celui-ci existe déjà normalement.

Code : Console

```
mateo21@mateo21-desktop:/usr/share/nano$ cd
mateo21@mateo21-desktop:~$ nano .bashrc
```

Édition du `.bashrc` personnel



Le fichier `.bashrc` est un peu complexe pour les simples mortels que nous sommes (pour le moment), donc attention à ne pas éditer n'importe quoi au risque de tout casser.
Bref, soyez juste un peu attentifs et tout ira bien.

Nous n'allons pas nous intéresser au `.bashrc` en détail. Nous allons seulement voir quelques lignes faciles à éditer qui vous permettront de personnaliser un peu votre console.

Personnaliser l'invite de commandes

Le fichier `.bashrc` vous permet entre autres choses de personnaliser l'invite de commandes. Vous savez, ce petit message qui s'affiche devant votre curseur dans la console :

Code : Console

```
mateo21@mateo21-desktop:~$
```

Rendez-vous plus bas dans le fichier, jusqu'à ce que vous tombiez sur ces lignes :

Code : Console

```
# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
xterm-color)
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[
    ;;
*)
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
```

```
    ;;
esac

# Comment in the above and uncomment this below for a color prompt
# PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01
```

Dans les commentaires, on vous dit que vous pouvez activer l'invite de commandes colorée en commentant les lignes du dessus et en décommentant la dernière ligne.
Rajoutez donc un # devant les deux premiers `PS1`, et enlevez le # devant le dernier `PS1` pour que la coloration de l'invite de commandes puisse fonctionner :

Code : Console

```
# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
xterm-color)
#     PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033
    ;;
*)
#     PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
    ;;
esac

# Comment in the above and uncomment this below for a color prompt
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;3
```

Enregistrez. Il faudra ouvrir une nouvelle console pour que la modification soit prise en compte afin de profiter d'une invite de commandes en couleurs.

Si vous êtes en forme, vous pouvez éditer la ligne que vous venez de décommenter :

Code : Console

```
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]: \[\033[01;
```

C'est en fait elle qui indique ce que l'invite de commandes doit afficher. Les séquences de type « `\033` » servent à paramétrer la couleur (ce n'est pas simple, je vous l'accorde).
Le symbole `\u` au milieu indique le nom de l'utilisateur (mateo21 par exemple) et `\h` indique le nom de la machine hôte (mateo21-desktop). Vous pouvez repérer dans cette ligne le « @ » qui sépare les deux, le « : », le « $ », etc.

Vous pouvez essayer de personnaliser un peu ces éléments ainsi que leur ordre si ça vous amuse (mais faites quand même attention à ne pas mettre le bazar là-dedans, hein. ;-).

Créer des alias

Les alias sont des commandes que vous créez et qui sont automatiquement transformées en d'autres commandes.
Descendez un peu plus bas dans le fichier, vous trouverez des lignes commentées commençant par « alias ».

Je vous invite à les personnaliser comme moi pour commencer :

Code : Console

```
# enable color support of ls and also add handy aliases
if [ "$TERM" != "dumb" ]; then
```

```
    eval "`dircolors -b`"
    alias ls='ls --color=auto'
    #alias dir='ls --color=auto --format=vertical'
    #alias vdir='ls --color=auto --format=long'
fi


# some more ls aliases
#alias ll='ls -l'
#alias la='ls -A'
#alias l='ls -CF'
```

Vous avez déjà probablement un alias créé :

Code : Console

```
alias ls='ls --color=auto'
```

Celui-ci active la coloration des résultats d'un `ls` à chaque fois que vous tapez `ls`. En fait, `ls` est systématiquement et automatiquement transformé par la console en `ls --color=auto`. C'est quand même plus rapide que de réécrire sans cesse ces paramètres.

Il y a un autre alias que j'ai l'habitude d'utiliser, c'est `ll` (deux fois la lettre L minuscule). Cela permet de faire un `ls` en mode détaillé.
Personnellement, j'ai un peu complété l'alias pour utiliser plus d'options à la fois, comme j'en ai parlé dans le chapitre sur `ls` :

Code : Console

```
alias ll='ls -lArth'
```

… signifie que la commande `ll` fera appel à `ls` avec les options qui permettent d'afficher le détail de chaque fichier, d'afficher les fichiers cachés, d'afficher les fichiers dans l'ordre inverse de dernière modification (le fichier le plus récent sera en bas) et d'afficher des tailles de fichiers lisibles pour un humain (`-h`).

La commande `ls` appellera automatiquement l'alias `ls --color=auto`, ce qui fait qu'un `ll` sera aussi coloré. Bref, c'est un peu un alias en chaîne.



Si vous tapez la commande `alias` dans la console, vous verrez la liste de tous les alias définis pour votre utilisateur.

Vous pouvez vous aussi définir vos propres alias. Comme vous pouvez le voir, c'est très simple car cela fonctionne sur le modèle :

Code : Console

```
alias nom='commande'
```

Attention à ne pas mettre d'espace autour du symbole « = ».

On peut par exemple en profiter pour sécuriser un peu nos `rm` pour éviter que l'on puisse supprimer tout le système depuis la racine `/`. Il y a en effet un paramètre de sécurité disponible avec `rm` : `--preserve-root`. Mais ce serait un peu long de l'écrire à chaque fois et on risquerait surtout d'oublier. En définissant un alias sur `rm`, vous ne pourrez pas oublier :

Code : Console

```
alias rm='rm --preserve-root'
```



Ne testez pas l'efficacité de cette commande en faisant un `rm -rf /` en root ! En effet, il faut relancer une console pour que les modifications soient prises en compte, et si vous avez fait une faute de frappe dans votre alias, vous ne serez pas protégés… mais pendant ce temps votre système sera détruit, lui !
Bref, même pour « vérifier », ne vous amusez pas à utiliser le `rm` de la mort…

Édition du `bashrc` global

Si vous voulez définir des alias ou modifier l'invite de commandes pour tous vos utilisateurs, vous pouvez le faire en une seule fois en éditant le fichier `bashrc` global situé dans : `/etc/bash.bashrc`.
Ce `bashrc` doit être édité en root.

Ce fichier propose un peu moins d'exemples commentés que celui présent dans votre home. Vous pouvez y copier vos alias et la ligne définissant l'invite de commandes (commençant par `PS1`).



Les éléments du `bashrc` personnel ont la priorité sur ceux du `bashrc` global. Si un même alias est défini dans les deux, c'est celui du `bashrc` personnel qui sera pris en compte.

Et aussi… le `.profile`

De même qu'il existe un ~/.bashrc et un `/etc/bash.bashrc`, il existe un ~/.profile et un `/etc/profile`. Quelle est la différence ?

En gros, le **.profile** est lu à chaque nouvelle console dans laquelle vous vous loggez (vous rentrez votre login / mot de passe). C'est le cas des consoles que vous ouvrez avec `Ctrl + Alt + F1` à `F6` (`tty1` à `tty6`).
Le **.bashrc** est lu lorsque vous ouvrez une console dans laquelle vous ne vous loggez pas. C'est le cas des consoles que vous ouvrez en mode graphique (Terminal sous Unity, Konsole sous KDE).

C'est un peu compliqué pour pas grand-chose au final. Dans la pratique, le `.profile` fait par défaut appel au `.bashrc`… Donc il suffit d'éditer votre `.bashrc` et vous modifierez ainsi les options de toutes vos consoles : celles avec et sans login. Voilà pourquoi je vous ai fait éditer dès le début le `.bashrc`. ;-)

Pour bien comprendre comment ça fonctionne, tout est résumé dans le schéma de la figure suivante.

Shell avec login :

Login / Mot de passe

Shell configuré par .profile

.profile

utilise

Shell sans login :

Shell configuré par .bashrc

.bashrc

Le fichier .profile appelle le .bashrc



Le shell est le programme qui interprète les commandes que vous tapez (vous pouvez considérer que c'est un synonyme de « console »).

On a, sur ce schéma, deux types de shell possibles :

- soit on a ouvert un shell qui demande un mot de passe et dans ce cas, c'est le `.profile` qui est lu pour la configuration ;
- soit on a ouvert un shell qui ne demande pas de mot de passe (c'est le cas d'une console en mode graphique en général) et dans ce cas-là, c'est le `.bashrc` qui servira à la configuration.

La particularité, comme le montre le schéma, c'est que le `.profile` fait appel au `.bashrc`… ce qui signifie que vous pouvez faire toutes vos configurations dans le `.bashrc` pour qu'elles soient valables quel que soit le type de shell que vous ouvrez.

En résumé

- Un éditeur de texte est un programme qui ouvre des fichiers texte (un peu comme Bloc-Notes sous Windows). On en a régulièrement besoin sous Linux pour modifier des fichiers de configuration, par exemple.
- Il existe de nombreux éditeurs de texte en console qui peuvent être très complets, comme Vim et Emacs.
- L'éditeur Nano est un des éditeurs en console les plus simples à utiliser ; nous commençons donc par découvrir celui-ci.
- On utilise plusieurs raccourcis clavier dans un éditeur de texte comme Nano. `Ctrl + W` lance une recherche, `Ctrl + O` enregistre le fichier, `Ctrl + X` permet de quitter, etc.
- On peut utiliser Nano pour modifier son fichier de configuration `.bashrc` et personnaliser sa console. On peut notamment s'en servir pour colorer l'invite de commandes et créer des alias.

 Installer des programmes avec apt-get

Quandvous êtes sous Windows et que vous voulez télécharger un nouveau programme, que faites-vous ?
En général une petite recherche sur un moteur de recherche, un tour sur les sites de téléchargement comme `telecharger.com`, `clubic.com`… et vous trouvez votre bonheur. Vous récupérez un programme d'installation, vous faites « Suivant », « Suivant », « Suivant », « Terminer » et c'est installé. Parfois, il faut répondre à des questions un peu techniques comme « Dans quel répertoire voulez-vous installer ce programme ? ».

Sous Linux (et notamment sous Ubuntu), ça ne fonctionne pas du tout comme ça : c'est encore plus simple. Mieux : vous allez vite vous rendre compte que c'est un vrai plaisir d'installer de nouveaux programmes et que c'est même un des points forts d'un système comme Ubuntu par rapport à Windows.
Vous allez voir.

Les paquets et leurs dépendances

Tout d'abord, il faut savoir que ce dont je vais vous parler ici concerne uniquement les distributions Linux basées sur Debian (je rappelle qu'Ubuntu en fait partie).
En effet, l'installation de programmes fonctionne différemment d'une distribution à une autre. C'est justement une des différences majeures entre les distributions.

Des programmes livrés sous forme de paquets

Sous Windows, vous connaissez ce que l'on appelle des « Programmes d'installation ». En général, il s'agit de fichiers `.exe` à lancer qui s'exécutent et extraient les fichiers du programme dans un dossier `Program Files`.
Exemple : le programme d'installation du jeu Trackmania Nations sous Windows est présenté par la figure suivante.



Sous Ubuntu, on n'a pas de programmes d'installation ; on a ce qu'on appelle des **paquets**.
Un paquet est une sorte de dossier zippé qui contient tous les fichiers du programme. Il se présente sous la forme d'un fichier `.deb`, en reférence à **DEB**ian. Il contient toutes les instructions nécessaires pour installer le programme.



Mais alors… un paquet `.deb`, c'est un peu comme un programme d'installation `.exe` sous Windows, non ?

Ça y ressemble, mais en fait ça fonctionne très différemment. Je citerai deux différences notables :

- il y a une gestion des **dépendances** du programme ;
- on n'a pas besoin de faire une recherche sur un moteur de recherche pour trouver un `.deb`. Tous les `.deb` sont rassemblés au même endroit sur un même serveur appelé **dépôt** (*repository*).

Ces deux points méritent plus d'explications.
Dans un premier temps nous allons voir ce que sont les dépendances ; dans un second temps, nous traiterons les dépôts.

Les dépendances, un cauchemar ?

Il est très rare qu'un programme puisse fonctionner seul sous Linux. Très souvent, il utilise d'autres programmes ou d'autres « bouts de programmes » appelés **bibliothèques**. On dit que les programmes dépendent d'autres programmes pour fonctionner : ils ont des **dépendances**.

Par exemple, le programme de dessin The GIMP (équivalent de Photoshop) ne peut pas fonctionner seul. Il dépend de bibliothèques de lecture des images (qui lui disent comment lire une image JPEG) par exemple. Parfois, ces dépendances ont elles-mêmes des dépendances !



Vous avez certainement déjà rencontré un peu ce problème sous Windows. Par exemple, quand vous installez un jeu, on vous dit « Il faut installer Direct X avant de pouvoir jouer à ce jeu ».

Heureusement, le système de paquets Debian est intelligent. Chaque paquet indique de quels autres paquets il dépend. Cela permet au système d'aller récupérer les dépendances manquantes automatiquement si besoin est.
Du coup, vous n'avez plus qu'à dire « Je veux installer Nano » et le système ira chercher toutes les dépendances manquantes tout seul !

Les dépôts

Comme je vous l'ai dit un peu plus haut, tous les paquets sont regroupés au sein d'un même endroit appelé dépôt. Il s'agit d'un serveur qui propose **tous** les paquets qui existent (ou presque), ce qui simplifie grandement vos recherches.

Sous Windows, les programmes sont éparpillés aux quatre coins du Net.
Sous Linux, on a décidé de ne pas refaire la même erreur. On a choisi de mettre tout le monde d'accord et de placer tous les programmes (paquets) au même endroit.



Ceci est possible en partie grâce au fait que la plupart des programmes sous Linux sont libres. En effet, les programmes étant « libres », tout le monde est autorisé à les diffuser ; ça ne pose donc aucun problème de les voir tous rassemblés au même endroit.
Sous Windows, la majorité des programmes étant propriétaire, leurs auteurs n'auraient jamais donné leur accord pour que ceux-ci soient diffusés au même endroit (gratuitement, qui plus est).

La notion de dépôt

L'endroit où tous les paquets se trouvent est appelé **dépôt** (*repository* en anglais).



Si tout le monde va chercher ses paquets sur un même dépôt, ça ne risque pas d'engorger le pauvre serveur qui les distribue ?

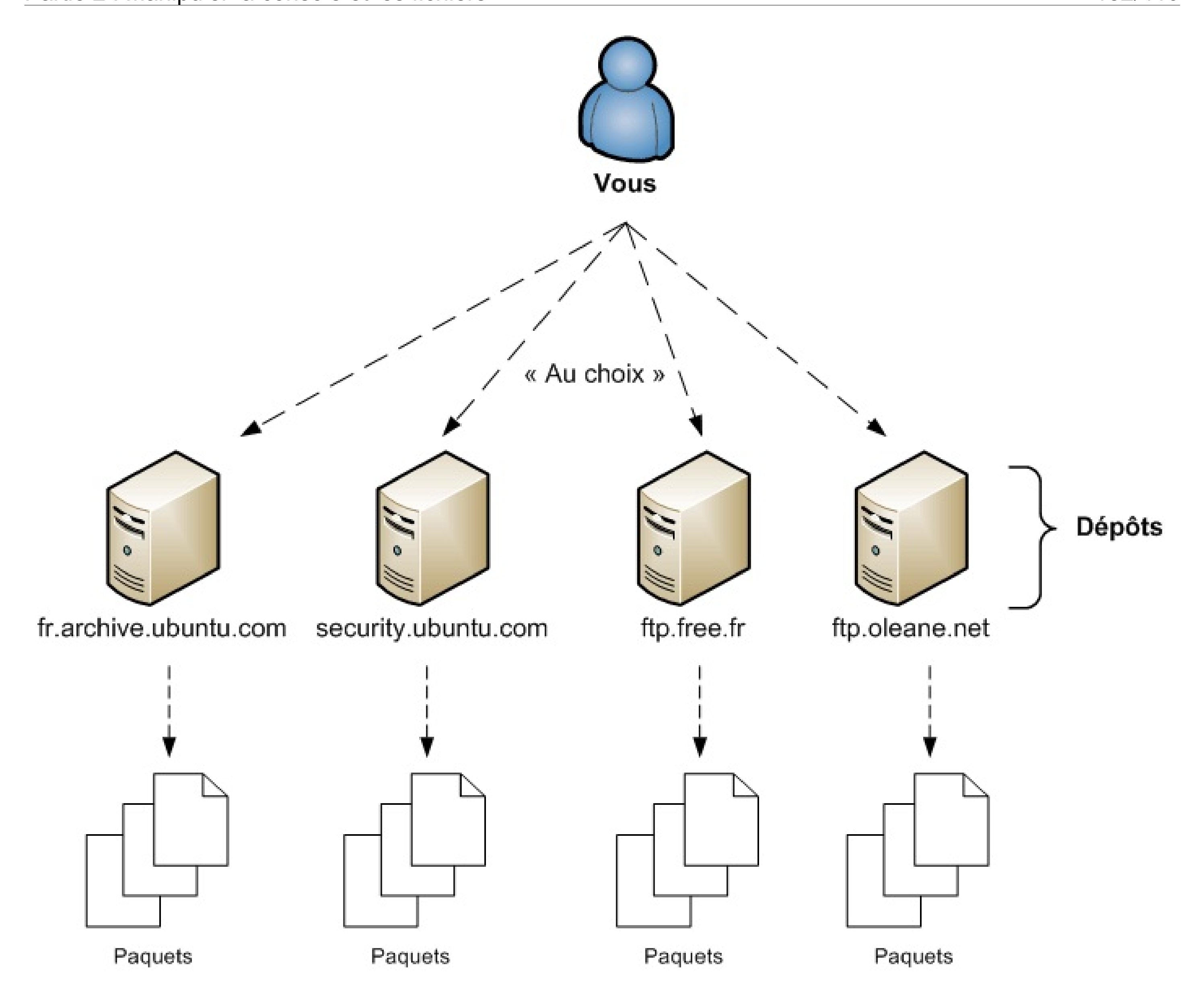
Bonne remarque, vous avez tout à fait raison. On peut certes mettre un gros serveur avec une grosse bande passante (qui permet à plusieurs centaines de personnes de télécharger en même temps), mais on peut difficilement imaginer que tous les linuxiens de la planète aillent se servir au même endroit au même moment !

C'est pour cela qu'il existe en fait un grand nombre de dépôts. La plupart proposent exactement les mêmes paquets (les dépôts sont donc des copies les uns des autres).
Certains dépôts spéciaux proposent toutefois des programmes que l'on ne trouve nulle part ailleurs, mais il est rare que l'on ait besoin de se servir dans ces dépôts.



En règle générale, avoir un seul « bon » dépôt suffit à installer la plupart des programmes qui existent sous Linux !

Schématiquement, je représenterais les dépôts comme sur la figure suivante.



C'est donc à vous de choisir le dépôt que vous voulez utiliser. Chacun de ces dépôts est identique : peu importe celui que vous choisissez, vous devriez retrouver les mêmes paquets.

Comme vous allez probablement beaucoup télécharger depuis votre dépôt, il est conseillé de choisir un serveur qui soit proche de chez vous (sur lequel vous téléchargez suffisamment vite).

En France, par défaut, Ubuntu utilise le dépôt `fr.archive.ubuntu.com`. Ce n'est pas toujours une bonne idée de garder le dépôt par défaut car en cas de nouvelle version d'Ubuntu et de ses logiciels, celui-ci est surchargé et devient alors très lent.

Si vous êtes chez Free, je vous recommande d'utiliser le dépôt de Free.
Si vous êtes chez Wanadoo / Orange, je vous recommande d'utiliser le dépôt Oleane (appartenant à Orange).

Nous allons voir comment changer de dépôt.

Gérer ses dépôts

Par défaut, quand vous installez Ubuntu, celui-ci utilise les dépôts officiels de la distribution. Seulement, comme je vous l'ai dit plus tôt, ces serveurs risquent d'être souvent encombrés.

Il n'y a aucun mal à utiliser les dépôts officiels, mais il peut être bien de savoir en changer. D'ailleurs, la liste des dépôts que votre ordinateur utilise est stockée dans un fichier. Pour éditer ce fichier, il faut utiliser un éditeur de texte comme… Nano, que l'on a vu au chapitre précédent justement (j'avais tout calculé, qu'est-ce que vous croyez ?).

C'est le moment de mettre en pratique ce que vous avez appris au chapitre précédent. Le fichier à ouvrir qui contient la liste des

dépôts que vous utilisez est :

`/etc/apt/sources.list`

Ce fichier ne peut être modifié que par root, l'administrateur de la machine.
Pour que vous puissiez modifier ce fichier, il faut donc passer root.

Pour cela, vous avez deux possibilités. Soit vous faites un `sudo` juste avant :

Code : Console

```
sudo nano /etc/apt/sources.list
```

… et vous modifierez le fichier en tant que root.
Soit vous passez root « définitivement » en faisant `sudo su` d'abord.

Le fichier que vous avez devrait ressembler à ceci :

Code : Console

```
# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.

deb http://fr.archive.ubuntu.com/ubuntu/ hardy main restricted
deb-src http://fr.archive.ubuntu.com/ubuntu/ hardy main restricted

## Major bug fix updates produced after the final release of the
## distribution.
deb http://fr.archive.ubuntu.com/ubuntu/ hardy-updates main restricted
deb-src http://fr.archive.ubuntu.com/ubuntu/ hardy-updates main restricted

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## universe WILL NOT receive any review or updates from the Ubuntu security
## team.
deb http://fr.archive.ubuntu.com/ubuntu/ hardy universe
deb-src http://fr.archive.ubuntu.com/ubuntu/ hardy universe

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## multiverse WILL NOT receive any review or updates from the Ubuntu
## security team.
deb http://fr.archive.ubuntu.com/ubuntu/ hardy multiverse
deb-src http://fr.archive.ubuntu.com/ubuntu/ hardy multiverse

## Uncomment the following two lines to add software from the 'backports'
## repository.
## N.B. software from this repository may not have been tested as
## extensively as that contained in the main release, although it includes
## newer versions of some applications which may provide useful features.
## Also, please note that software in backports WILL NOT receive any review
## or updates from the Ubuntu security team.
# deb http://fr.archive.ubuntu.com/ubuntu/ hardy-
backports main restricted universe multiverse
# deb-src http://fr.archive.ubuntu.com/ubuntu/ hardy-
backports main restricted universe multiverse

deb http://security.ubuntu.com/ubuntu hardy-security main restricted
deb-src http://security.ubuntu.com/ubuntu hardy-security main restricted
deb http://security.ubuntu.com/ubuntu hardy-security universe
deb-src http://security.ubuntu.com/ubuntu hardy-security universe
deb http://security.ubuntu.com/ubuntu hardy-security multiverse
deb-src http://security.ubuntu.com/ubuntu hardy-security multiverse
```

Les lignes commençant par un # sont des lignes de commentaires. Elles seront ignorées.
Normalement, chaque ligne du fichier commence par une de ces deux directives :

- **deb** : pour télécharger la version compilée (binaire) des programmes. C'est ce que vous voudrez faire dans la plupart des cas car c'est la version « prête à l'emploi » ;
- **deb-src** : permet de récupérer le code source du programme. Généralement, vous n'en avez pas besoin, sauf si vous êtes curieux et que vous voulez voir la source d'un programme. C'est l'avantage des logiciels libres de pouvoir consulter la source des programmes !

A priori seules les lignes `deb` nous intéressent. On pourrait même supprimer (ou commenter) les lignes `deb-src`, récupérer les sources n'ayant aucun intérêt pour nous.

Voici une ligne « type » :

```
deb http://fr.archive.ubuntu.com/ubuntu/ hardy universe
```

En premier paramètre, on a l'adresse du dépôt. Ici, le dépôt français par défaut est `http://fr.archive.ubuntu.com/ubuntu/`.
Ensuite, on a le nom de la version de la distribution qu'on utilise, « hardy » dans ce cas.
Enfin, le dernier paramètre (et tous les paramètres suivants s'il y en a) correspond à la « section » du dépôt dans laquelle vous voulez regarder.

Ouf… c'est un peu compliqué, tout ça.
En fait, la seule chose que vous devriez avoir à faire, c'est remplacer toutes les adresses (http…) par celle du nouveau dépôt que vous voulez utiliser.



Mais comment je connais l'adresse des autres dépôts qui existent ?

Bonne question… à laquelle je ne peux pas répondre.
A priori tout le monde peut créer un dépôt, il peut donc très bien exister des dizaines, voire des centaines de dépôts différents que vous pourriez utiliser. Je ne les connais pas tous et je ne vais pas me risquer à dresser une liste ici, mais si vous recherchez « dépôts ubuntu » avec votre moteur de recherche favori, vous devriez trouver des réponses.

Utiliser l'outil graphique

Le plus simple, je pense, est de passer par l'outil graphique fourni par Ubuntu. L'outil en question dépend de votre Ubuntu. Si vous avez :

- **Ubuntu** : allez dans `Système` → `Administration` → `Sources de logiciels` ;
- **KUbuntu** : allez dans `Menu K` → `Système` → `Gestionnaire Adept` → `Adept` → `Gérer les dépôts` ;
- **XUbuntu** : allez dans `Applications` → `Système` → `Sources de mises à jour`.

Par exemple, la fenêtre sous Xubuntu est présentée sur la figure suivante.

Sources de mise à jour

Ubuntu Logiciel | Logiciels de tierces parties | Mises à jour | Authentification | Statistiques

Téléchargeable depuis Internet

- [x] Logiciels libres pris en charge par Canonical (main)
- [x] Logiciels libres maintenus par la communauté (universe)
- [x] Pilotes propriétaires de périphériques (restricted)
- [x] Logiciel soumis au droit d'auteur ou à des restrictions légales (multiverse)
- [ ] Code source

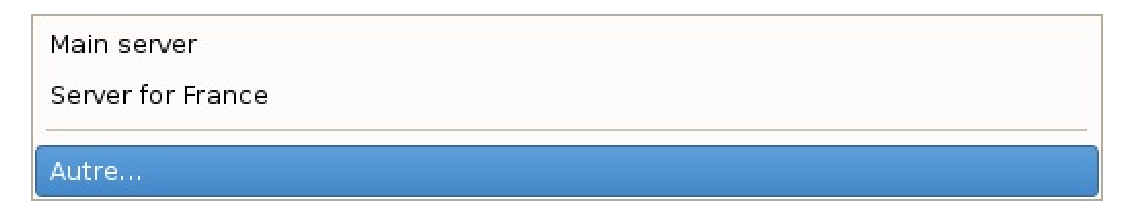
Télécharger depuis : Server for France

Installable à partir du CD-ROM/DVD

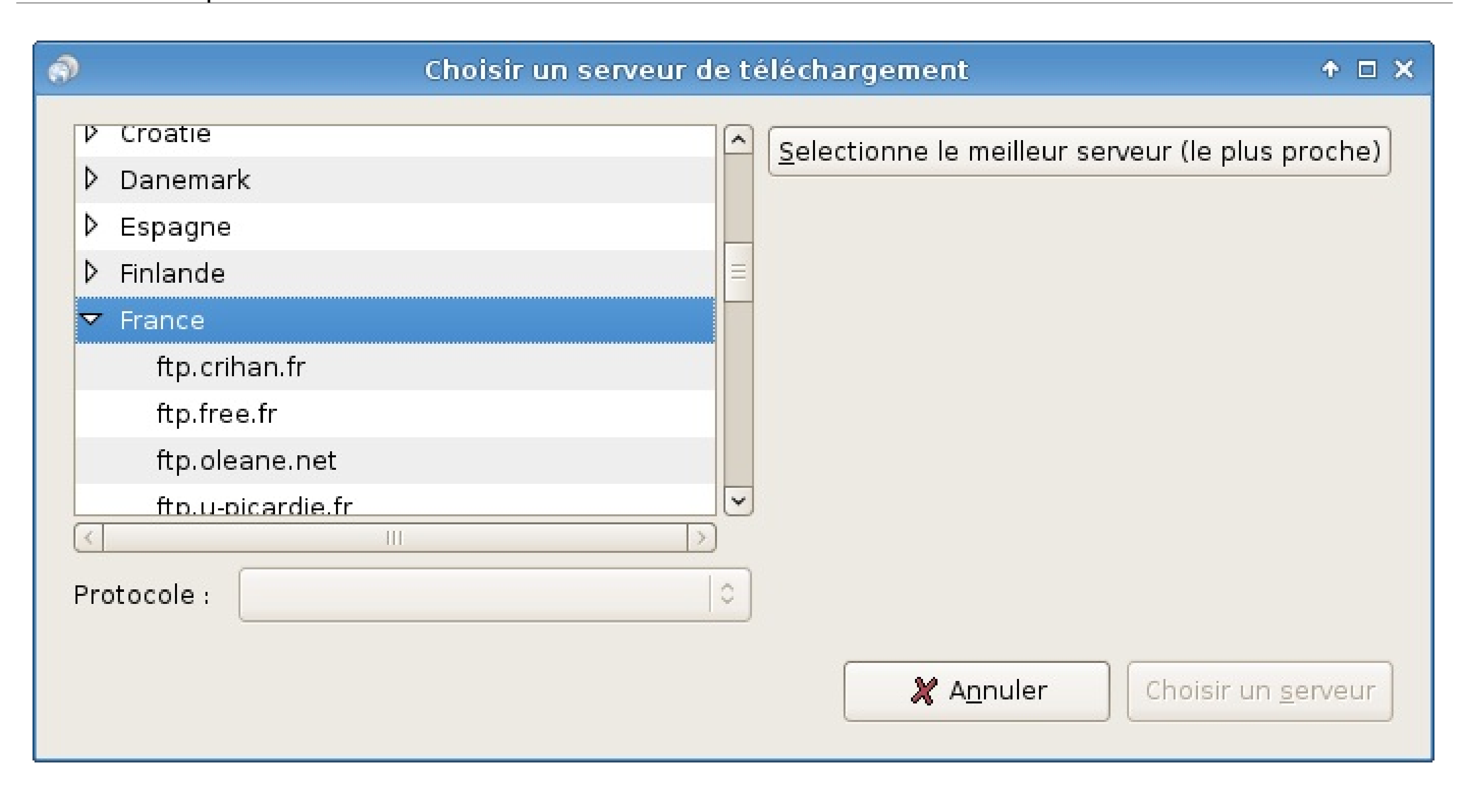
Pour installer depuis un CD-ROM ou un DVD, insérez le disque dans le lecteur.

Rétablir | Fermer

Notez la liste déroulante « Télécharger depuis : Server for France », qui signifie que vous utilisez les dépôts français officiels d'Ubuntu.
Cette liste vous offre le choix pour le moment entre « Main server » (le serveur américain officiel) et le serveur français. Cliquez sur « Autre… » (figure suivante).



Cette fenêtre recense les dépôts les plus connus regroupés par pays (figure suivante). Certains sont fournis par des universités (u-picardie.fr), d'autres par des FAI (Free, Oleane pour Orange…).



Pour savoir lequel est le plus rapide, cliquez sur le bouton « Sélectionner le meilleur serveur (le plus proche) ». Une fenêtre va s'ouvrir et tester la vitesse de tous les serveurs (figure suivante).



Le serveur qui sera sélectionné à la fin sera celui que le logiciel aura détecté comme étant le plus rapide pour vous. Cliquez sur « Choisir un serveur » pour valider.

Ensuite, le logiciel vous fera remarquer que votre cache n'est pas à jour. En effet, pour des raisons de performances, Ubuntu avait téléchargé la liste des paquets proposés par l'ancien serveur. Comme vous venez d'en changer, il se peut que la liste des paquets ait changé elle aussi. Il faut récupérer la liste des paquets proposés par le nouveau serveur et la mettre en cache.
Cliquez sur « Actualiser » pour mettre à jour la liste des logiciels disponibles (figure suivante).



Et voilà !
L'outil graphique est plus pratique pour mettre à jour votre fichier `sources.list`, il faut bien le reconnaître.

Maintenant que nous avons choisi notre dépôt, nous sommes prêts à télécharger à gogo !

Les outils de gestion des paquets

Résumons.
Jusqu'ici, nous avons découvert trois nouveaux termes :

- **paquet** : c'est un programme « prêt à l'emploi », l'équivalent des programmes d'installation sous Windows en quelque sorte ;
- **dépendance** : un paquet peut avoir besoin de plusieurs autres paquets pour fonctionner, on dit qu'il a des dépendances ;
- **dépôt** : c'est le serveur sur lequel on va télécharger nos paquets.

Je vous ai proposé de changer de dépôt mais sachez que ce n'est pas obligatoire, vous pouvez vous contenter de celui par défaut.

Sous Ubuntu, on peut utiliser un programme graphique qui gère les paquets pour nous : nous avions justement découvert la logithèque Ubuntu au début de ce cours.

Ici, nous nous intéressons aux manipulations en console. Les deux programmes console de gestion des paquets les plus connus sont :

- `apt-get` ;
- `aptitude`.

Lequel des deux utiliser ?
Le premier est sûrement le plus célèbre ; le second est généralement reconnu comme étant plus efficace lors de la désinstallation de paquets (il supprime aussi les dépendances inutilisées). Cependant `apt-get` sous Ubuntu a évolué aussi et peut supprimer les dépendances inutilisées.

Après, c'est un peu une question d'habitude. Pour ma part, j'ai toujours été habitué à `apt-get`, c'est donc celui que j'utilise et que je vais vous montrer. Que vous utilisiez l'un ou l'autre ne fera pas beaucoup de différence.

Nous devons généralement suivre trois étapes pour télécharger un paquet :

- **`apt-get update`** (optionnel) : pour mettre notre cache à jour si ce n'est pas déjà fait ;
- **`apt-cache search monpaquet`** (optionnel) : pour rechercher le paquet que nous voulons télécharger si nous ne connaissons pas son nom exact ;
- **`apt-get install monpaquet`** : pour télécharger et installer notre paquet.

C'est très simple, vous allez voir.
Nous verrons aussi comment supprimer un paquet et comment mettre tous nos paquets à jour en une seule commande !

apt-get update : mettre à jour le cache des paquets

Commençons par la mise à jour du cache des paquets (`apt-get update`).
Cela correspond à télécharger la nouvelle liste des paquets proposés par le dépôt.

Toutefois, il n'est pas nécessaire de mettre à jour son cache à chaque fois que l'on veut télécharger un paquet.



Ah bon ? Comment je sais si je dois mettre à jour mon cache, alors ?

Il y a deux cas où vous avez besoin de le mettre à jour :

- quand vous changez ou ajoutez un dépôt à votre liste de dépôts ;
- quand vous n'avez pas mis à jour votre cache depuis un moment (quelques semaines).

Pour mettre à jour votre cache, tapez ceci dans la console **en tant que root** :

Code : Console

```
apt-get update
```



Pensez à rajouter un `sudo` si vous n'êtes pas déjà root.

Après avoir tapé cette commande, vous allez automatiquement télécharger la dernière liste des paquets proposés par vos dépôts :

Code : Console

```
root@mateo21-desktop:~# apt-get update
Réception de : 1 http://wine.budgetdedicated.com hardy Release.gpg [191B]
Ign http://wine.budgetdedicated.com hardy/main Translation-fr
Atteint http://wine.budgetdedicated.com hardy Release
Atteint ftp://ftp.free.fr hardy Release.gpg
Ign http://wine.budgetdedicated.com hardy/main Packages
Atteint ftp://ftp.free.fr hardy/restricted Translation-fr
Atteint http://wine.budgetdedicated.com hardy/main Sources
Atteint ftp://ftp.free.fr hardy/main Translation-fr
Atteint http://wine.budgetdedicated.com hardy/main Packages
Atteint ftp://ftp.free.fr hardy/universe Translation-fr
Atteint ftp://ftp.free.fr hardy/multiverse Translation-fr
Atteint ftp://ftp.free.fr hardy-updates Release.gpg
Réception de : 2 ftp://ftp.free.fr hardy-updates/restricted Translation-fr
Ign ftp://ftp.free.fr hardy-updates/restricted Translation-fr
Réception de : 3 ftp://ftp.free.fr hardy-updates/main Translation-fr
Ign ftp://ftp.free.fr hardy-updates/main Translation-fr
Réception de : 4 ftp://ftp.free.fr hardy-security Release.gpg [191B]
Réception de : 5 ftp://ftp.free.fr hardy-security/restricted Translation-fr
Ign ftp://ftp.free.fr hardy-security/restricted Translation-fr
Réception de : 6 ftp://ftp.free.fr hardy-security/main Translation-fr
Ign ftp://ftp.free.fr hardy-security/main Translation-fr
Réception de : 7 ftp://ftp.free.fr hardy-security/universe Translation-fr
Ign ftp://ftp.free.fr hardy-security/universe Translation-fr
Réception de : 8 ftp://ftp.free.fr hardy-security/multiverse Translation-fr
Ign ftp://ftp.free.fr hardy-security/multiverse Translation-fr
Réception de : 9 ftp://ftp.free.fr hardy Release [57,2kB]
Réception de : 10 ftp://ftp.free.fr hardy-updates Release [50,9kB]
Réception de : 11 ftp://ftp.free.fr hardy-security Release [50,9kB]
Atteint ftp://ftp.free.fr hardy/restricted Packages
Atteint ftp://ftp.free.fr hardy/main Packages
Atteint ftp://ftp.free.fr hardy/restricted Sources
Atteint ftp://ftp.free.fr hardy/universe Packages
Atteint ftp://ftp.free.fr hardy/universe Sources
Atteint ftp://ftp.free.fr hardy/multiverse Packages
Atteint ftp://ftp.free.fr hardy/multiverse Sources
Atteint ftp://ftp.free.fr hardy-updates/restricted Packages
Atteint ftp://ftp.free.fr hardy-updates/main Packages
Atteint ftp://ftp.free.fr hardy-updates/restricted Sources
```

```
Réception de : 12 ftp://ftp.free.fr hardy-security/restricted Packages [5990B]
Réception de : 13 ftp://ftp.free.fr hardy-security/main Packages [120kB]
Réception de : 14 ftp://ftp.free.fr hardy-security/restricted Sources [956B]
Réception de : 15 ftp://ftp.free.fr hardy-security/universe Packages [78,6kB]
Réception de : 16 ftp://ftp.free.fr hardy-security/universe Sources [11,8kB]
Réception de : 17 ftp://ftp.free.fr hardy-security/multiverse Packages [5395B]
Réception de : 18 ftp://ftp.free.fr hardy-security/multiverse Sources [1042B]
382ko réceptionnés en 7s (50,1ko/s)
Lecture des listes de paquets... Fait
```

Voilà : ça fait un peu peur la première fois mais en général, c'est assez rapide.
Je vous rappelle que vous n'avez pas besoin d'exécuter cette commande à chaque fois que vous voulez installer un paquet mais seulement de temps en temps pour être sûrs d'avoir la liste la plus à jour possible.

apt-cache search : rechercher un paquet

À moins que vous ne connaissiez déjà le nom exact du paquet que vous voulez, il va falloir effectuer une petite recherche.
On utilise pour cela la commande suivante :

Code : Console

```
apt-cache search votrerecherche
```

Cette commande effectue une recherche de paquet dans votre cache. Cela évite d'avoir à aller sur Internet pour faire la recherche, ce qui aurait été lent.

Bon ! Que veut-on télécharger ? Un éditeur de texte ? Un navigateur ?
Allez : pour s'amuser, je vous propose de rechercher un jeu, par exemple un jeu de casse-briques (*breakout* en anglais).

Faites la recherche suivante :

Code : Console

```
root@mateo21-desktop:~# apt-cache search breakout
briquolo - Fast paced 3d Breakout
briquolo-data - Fast paced 3d Breakout data files
circuslinux - The clowns are trying to pop balloons to score points!
circuslinux-data - data files for circuslinux
gnome-breakout - Clone of the classic game Breakout, written for GNOME
lbreakout2 - A ball-and-paddle game with nice graphics
lbreakout2-data - A ball-and-paddle game with nice graphics (DATA FILES)
libfreebob0 - FreeBoB API
libfreebob0-dev - FreeBoB API - development files
tecnoballz - breaking block game ported from the Amiga platform
```

La commande `apt-cache search breakout` a listé tous les paquets qui avaient un rapport avec les casse-briques.
À gauche vous avez le nom du paquet, à droite une courte description.


Si vous voulez une plus ample description d'un paquet, utilisez `apt-cache show nomdupaquet`. Exemple : `apt-cache show lbreakout2`.

apt-get install : installer un paquet

Pour ma part, j'aime beaucoup le jeu `lbreakout2` (figure suivante).
Je vous propose donc de le télécharger, ce qui se fait très simplement (**toujours en tant que root**, rajoutez un `sudo` devant la commande si vous n'êtes pas déjà root) :

Code : Console

```
apt-get install lbreakout2
```

La commande `apt-get install` attend que vous lui donniez le nom du paquet à installer.



Astuce : vous pouvez installer plusieurs paquets d'un coup en les listant un à un :
`apt-get install paquet1 paquet2 paquet3`

Essayons donc d'installer `lbreakout2` :

Code : Console

```
root@mateo21-desktop:~# apt-get install lbreakout2
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture de l'information d'état... Fait
Les paquets supplémentaires suivants seront installés :
  lbreakout2-data libsdl-mixer1.2 libsmpeg0
Les NOUVEAUX paquets suivants seront installés :
  lbreakout2 lbreakout2-data libsdl-mixer1.2 libsmpeg0
0 mis à jour, 4 nouvellement installés, 0 à enlever et 153 non mis à jour.
Il est nécessaire de prendre 2943ko dans les archives.
Après dépaquetage, 5358ko d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer [O/n] ? O
Réception de : 1 ftp://ftp.free.fr feisty/main libsmpeg0 0.4.5+cvs20030824-
1.9build1 [105kB]
Réception de : 2 ftp://ftp.free.fr feisty/main libsdl-mixer1.2 1.2.6-
1.1build1 [145kB]
Réception de : 3 ftp://ftp.free.fr feisty/universe lbreakout2-data 2.5.2-
```

```
2.1ubuntu1 [2444kB]
Réception de : 4 ftp://ftp.free.fr feisty/universe lbreakout2 2.5.2-
2.1ubuntu1 [249kB]
2943ko réceptionnés en 6s (484ko/s)
Sélection du paquet libsmpeg0 précédemment désélectionné.
(Lecture de la base de données... 123350 fichiers et répertoires déjà installés.)
Dépaquetage de libsmpeg0 (à partir de .../libsmpeg0_0.4.5+cvs20030824-
1.9build1_amd64.deb) ...
Sélection du paquet libsdl-mixer1.2 précédemment désélectionné.
Dépaquetage de libsdl-mixer1.2 (à partir de .../libsdl-mixer1.2_1.2.6-
1.1build1_amd64.deb) ...
Sélection du paquet lbreakout2-data précédemment désélectionné.
Dépaquetage de lbreakout2-data (à partir de .../lbreakout2-data_2.5.2-
2.1ubuntu1_all.deb) ...
Sélection du paquet lbreakout2 précédemment désélectionné.
Dépaquetage de lbreakout2 (à partir de .../lbreakout2_2.5.2-
2.1ubuntu1_amd64.deb) ...
Paramétrage de libsmpeg0 (0.4.5+cvs20030824-1.9build1) ...

Paramétrage de libsdl-mixer1.2 (1.2.6-1.1build1) ...

Paramétrage de lbreakout2-data (2.5.2-2.1ubuntu1) ...
Paramétrage de lbreakout2 (2.5.2-2.1ubuntu1) ...
```



Si vous obtenez le message d'erreur suivant :

```
E: Impossible d'ouvrir le fichier verrou /var/lib/dpkg/lock -
 open (13 Permission non accordée)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?
```

… c'est que vous n'avez pas exécuté la commande en tant que root. Pensez à passer root (en utilisant `sudo`) avant de l'exécuter.

Il s'en est passé des choses, dites donc !

Comme vous pouvez le voir, au début `apt-get` a vérifié si le paquet existait et les dépendances dont il avait besoin. Il se trouve que `lbreakout2` avait besoin de dépendances qu'on n'a pas, comme `libsdlmixer-1.2` par exemple.
C'est pour cela qu'on nous a demandé confirmation au milieu :

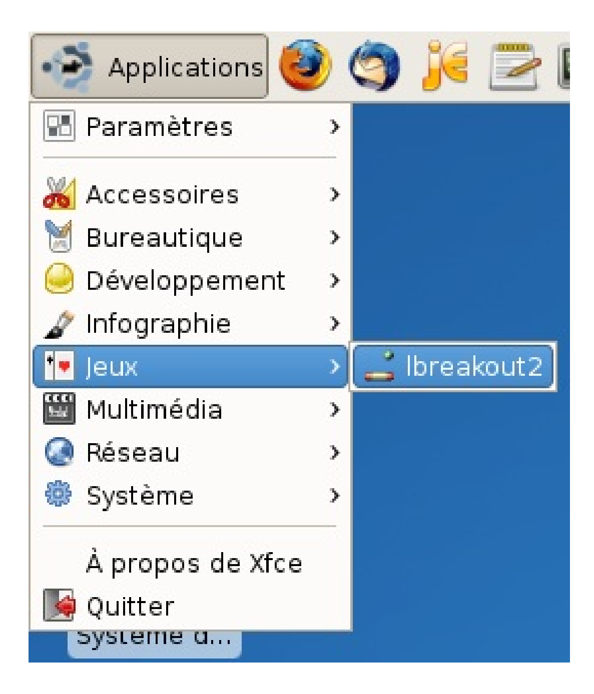
```
Souhaitez-vous continuer [O/n] ?
```

Répondez par un « O » majuscule (comme « Oui ») et tapez `Entrée` pour que l'installation se poursuive.

C'est alors que la magie d'`apt-get` opère : le programme va aller télécharger tout seul le paquet sur le dépôt ainsi que toutes les dépendances dont il a besoin et que nous n'avons pas.
Puis il « dépaquète » les fichiers qui étaient contenus à l'intérieur du paquet, les installe et effectue les paramétrages tout seul.

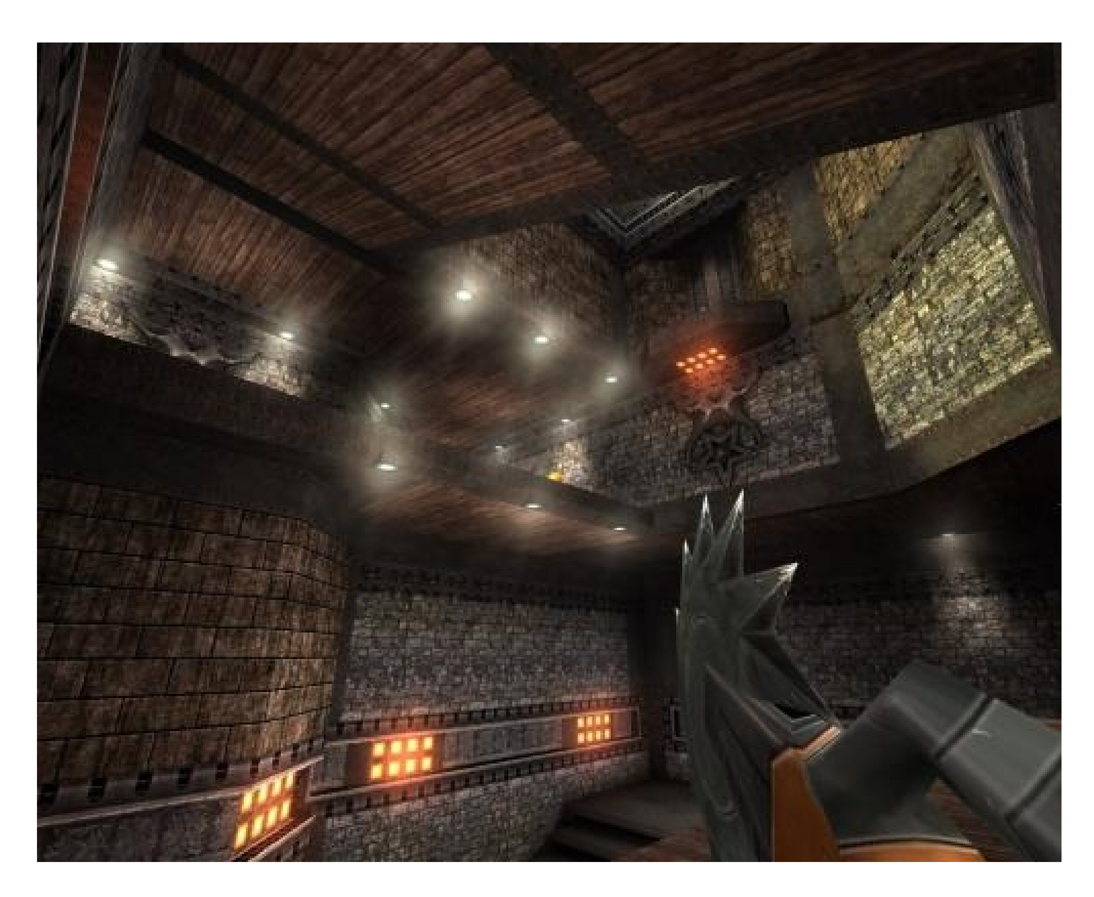
Ce qui est fort là-dedans, c'est qu'`apt-get` ne vous demande rien ! Il installe tout ce qu'il faut tout seul, dans les bons répertoires, et crée même le raccourci pour lancer le jeu dans la section « Jeux » !

Vérifiez par vous-mêmes. Le menu des applications d'Ubuntu comporte désormais une section `Jeux` (qu'il a créée si elle n'existait pas) et possède un raccourci vers le jeu que nous venons d'installer (figure suivante).



C'est bon, vous pouvez jouer !

Recommencez l'opération autant de fois que vous voulez, tous les paquets que vous devriez voir sont des logiciels libres que vous pouvez télécharger à volonté.
Sur la figure suivante, vous pouvez voir OpenArena, basé sur le célèbre jeu Quake III Arena dont le code source est devenu libre.



Pour l'obtenir, c'est très simple :

Code : Console

```
apt-get install openarena
```

C'est aussi simple que ça. Toutefois il est fortement conseillé d'avoir installé les pilotes de votre carte graphique avant d'y jouer.

apt-get autoremove : supprimer un paquet

Si vous voulez désinstaller un paquet, vous pouvez utiliser la commande `apt-get remove` :

Code : Console

```
apt-get remove lbreakout2
```

Le paquet sera alors désinstallé de votre ordinateur.
Toutefois, cela ne supprime pas les dépendances du paquet devenues inutiles. Pour demander à `apt-get` de supprimer **aussi** les dépendances inutiles, on utilise `autoremove` :

Code : Console

```
apt-get autoremove lbreakout2
```

Faisons un `autoremove`, comme ça nous serons sûrs de libérer un maximum d'espace disque.

Code : Console

```
root@mateo21-desktop:~# apt-get autoremove lbreakout2
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture de l'information d'état... Fait
Les paquets suivants ont été automatiquement installés mais ne sont plus nécessaire
  libsdl-mixer1.2 libsmpeg0
Les paquets suivants seront ENLEVÉS :
  lbreakout2 lbreakout2-data libsdl-mixer1.2 libsmpeg0
0 mis à jour, 0 nouvellement installés, 4 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 0o dans les archives.
Après dépaquetage, 5358ko d'espace disque seront libérés.
Souhaitez-vous continuer [O/n] ?
```

`apt-get` analyse le paquet, vérifie les dépendances qu'il peut supprimer sans risque (c'est-à-dire les dépendances qui ne sont plus utilisées par aucun autre paquet) et calcule la taille d'espace disque libéré (un peu plus de 5 Mo dans mon cas).

Faites « O » pour confirmer que vous voulez bien désinstaller le paquet et ses dépendances.
Vous voyez alors les paquets se faire désinstaller par `apt-get` :

Code : Console

```
Suppression de lbreakout2 ...
Suppression de lbreakout2-data ...
Suppression de libsdl-mixer1.2 ...
Suppression de libsmpeg0 ...
```

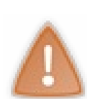
C'est fini, le paquet et ses dépendances sont proprement désinstallés. :-)

apt-get upgrade : mettre à jour tous les paquets

Une autre fonctionnalité particulièrement géniale d'`apt-get` est sa capacité à mettre à jour tous les paquets installés sur votre système d'un seul coup. Le programme ira chercher les nouvelles versions de tous vos programmes et les mettra à jour si une nouvelle version est disponible :

Code : Console

```
apt-get upgrade
```



Pensez à faire un `apt-get update` pour mettre à jour le cache des paquets sur votre machine avant de lancer un `upgrade`.
En effet, `apt-get` compare la version de vos paquets installés avec ceux présents dans le cache. Si votre cache est « ancien », `apt-get` se dira « Oh bah, il n'y a rien de nouveau, pas besoin d'une mise à jour ». Veillez donc à faire régulièrement un `apt-get update` pour être 100 % sûrs que votre cache est à jour.

On vous demandera une confirmation après avoir listé tous les paquets qui ont besoin d'une mise à jour.
Vous n'avez pas besoin de faire autre chose. Tous vos paquets installés seront mis à jour (ça peut être un peu long, par contre).

Si tous vos paquets sont déjà dans leur version la plus récente, vous verrez le message suivant :

Code : Console

```
root@mateo21-desktop:~# apt-get upgrade
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture de l'information d'état... Fait
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

Tous ces « 0 » de la dernière ligne nous indiquent que rien ne s'est passé car il n'y avait rien à faire.
Cela signifie que votre système est à jour.



Il est conseillé de faire un `apt-get upgrade` régulièrement pour avoir le système le plus à jour possible. Cela vous permet de bénéficier des dernières fonctionnalités des logiciels, mais cela corrige aussi les failles de sécurité qui auraient pu être découvertes dans les programmes (et on en trouve tous les jours, même dans les logiciels libres !).

En résumé

- La plupart des distributions Linux proposent un moyen centralisé de télécharger et d'installer des logiciels facilement. L'installation de programmes y est généralement beaucoup plus simple que sous Windows !
- Sous Ubuntu, on peut utiliser des interfaces graphiques pour télécharger et installer des logiciels. En console, on fait appel au programme `apt-get`.
- On télécharge les programmes depuis des serveurs (fournis par Ubuntu, votre fournisseur d'accès ou une université) qui font office de dépôts.
- `apt-get update` met à jour la liste des programmes (appelés *paquets*) qui existent.
- `apt-cache search` permet de rechercher dans la liste des paquets.
- `apt-get install` télécharge et installe un paquet.
- `apt-get upgrade` met à jour tous les paquets installés.
- `apt-get autoremove` permet de supprimer un paquet.



RTFM : lisez le manuel !

Quand on vient de Windows, on n'a pas trop l'habitude de lire des documentations. Parfois les logiciels sont livrés avec des modes d'emploi, mais honnêtement, qui ici prend la peine de les lire ?

Sous Linux, lire la documentation doit devenir un **réflexe**. En effet, bien que cela fasse un peu peur au premier abord, la documentation est vraiment le meilleur endroit pour en savoir plus sur les commandes que vous utilisez.

Les livres que j'écris ne pourront jamais rivaliser avec la documentation. Je peux vous montrer les commandes et les paramètres qui me semblent les plus utiles, mais pour connaître certains paramètres dont vous avez besoin moins souvent, vous n'y couperez pas : **vous aurez besoin de lire la doc**.
Ce chapitre est justement là pour « démystifier » le manuel et vous apprendre à le lire. C'est peut-être un des chapitres les plus importants du cours, car si vous savez lire la doc, vous êtes capables d'apprendre tout ce dont vous avez besoin… et vous pourrez donc tout faire !



Le titre de ce chapitre est une insulte « amicale » entre linuxiens. C'est en général ce qu'on dit à un débutant qui pose trop de questions sur les forums alors que la réponse se trouve dans la doc.
« RTFM » est une abréviation qui signifie « *Read The Fucking Manual* », ce qui se traduirait en français par quelque chose comme « Lis ce p*** de manuel ». Mais oui, c'est amical ! Puisque je vous le dis…

man : afficher le manuel d'une commande

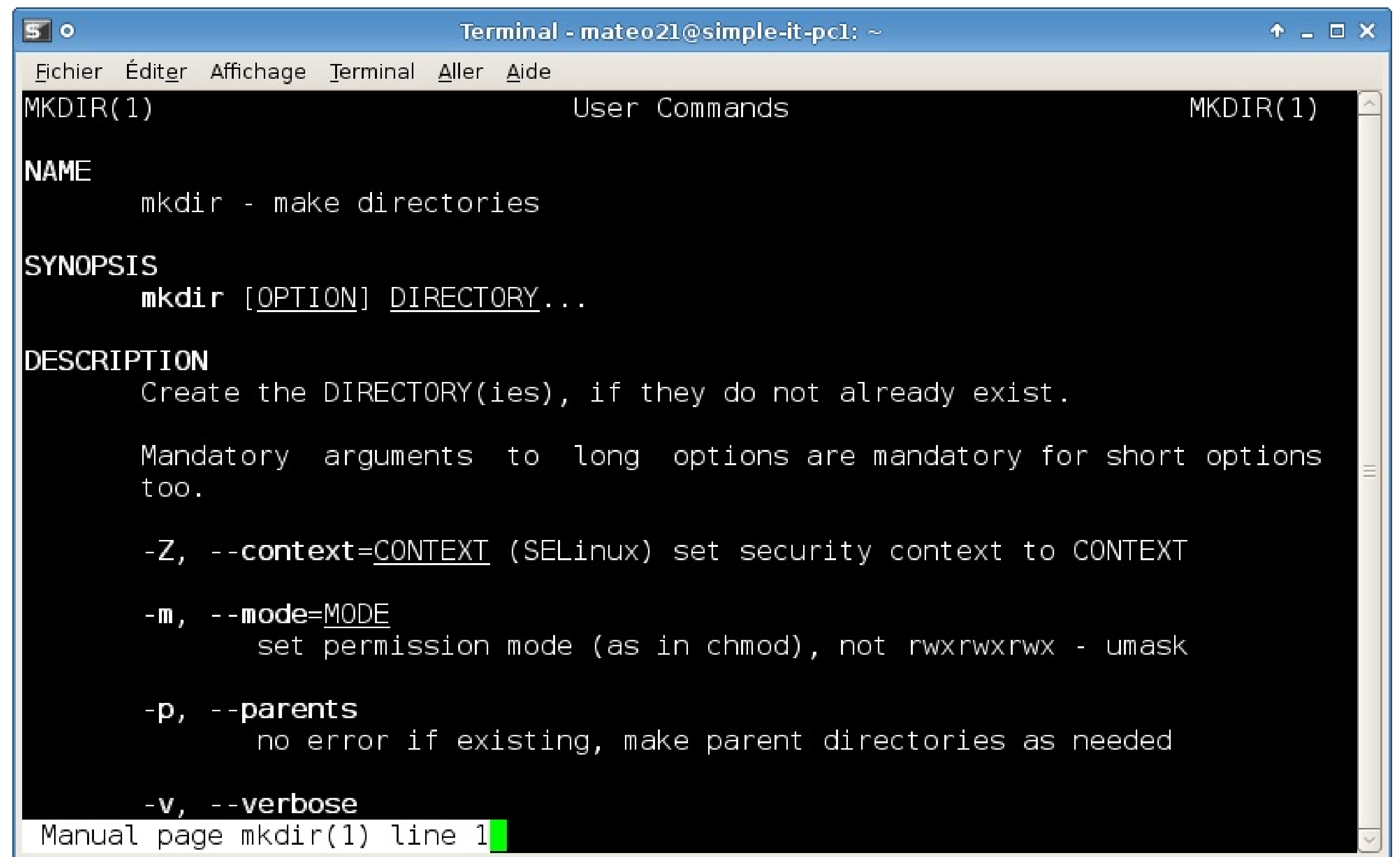
La commande magique que nous allons expérimenter tout au long de ce chapitre est `man`, qui est l'abréviation de ***manual***.

La commande `man` s'utilise très simplement : elle prend en paramètre le nom de la commande dont vous voulez lire la doc.
Par exemple, si je veux tout savoir sur la commande `mkdir`, je dois écrire :

Code : Console

```
man mkdir
```

Votre console devrait alors ressembler à l'image de la figure suivante.



Le manuel de la commande mkdir

Il s'agit de la documentation de la commande `mkdir`. Là-dedans, il y a tout ce qu'il faut savoir sur `mkdir`.
La doc de la commande étant généralement un peu longue, celle-ci s'affiche page par page, à la manière de `less` qu'on a vue dans un chapitre précédent.

Se déplacer dans le manuel

Voici quelques commandes à connaître pour se déplacer dans le manuel.

- Utilisez les **touches fléchées** du clavier (vers le haut et vers le bas) pour vous déplacer ligne par ligne.
- Vous pouvez utiliser les touches `Page Up` et `Page Down` (ou `Espace`) pour vous déplacer de page en page.
- Appuyez sur la touche `Home` (aussi appelée `Origine`) pour revenir au début du manuel, et sur `Fin` pour aller à la fin.
- Appuyez sur la touche `/` (slash) pour effectuer une recherche ; c'est très pratique ! Tapez ensuite le mot que vous recherchez dans le manuel puis appuyez sur `Entrée`. Si la recherche renvoie un résultat vous serez automatiquement placés sur le premier résultat trouvé. Pour passer au résultat suivant, tapez à nouveau `/` puis directement sur `Entrée` (sans retaper votre recherche).
- Appuyez sur la touche `Q` pour quitter le manuel à tout moment, comme vous le faisiez avec `less`.



Si comme moi vous avez ouvert votre manuel dans une console graphique, vous pouvez aussi utiliser la molette de la souris !

Les principales sections du manuel

Comme vous pouvez le voir, le manuel de la commande est découpé en plusieurs sections (leurs noms sont écrits en gras et alignés à gauche de l'écran).
Voici leur signification.

- `NAME` : le nom de la commande dont vous êtes en train d'afficher le manuel ainsi qu'une courte description de son utilité.
- `SYNOPSIS` : c'est la liste de toutes les façons d'utiliser la commande. Nous y reviendrons un peu plus loin car il est vital de comprendre ce qui est écrit dans cette section.
- `DESCRIPTION` : une description plus approfondie de ce que fait la commande. On y trouve aussi la liste des paramètres et leur signification. C'est en général la section la plus longue.
- `AUTHOR` : l'auteur du programme. Il y a parfois de nombreux auteurs ; c'est souvent le cas d'ailleurs avec le logiciel libre.
- `REPORTING BUGS` : si vous rencontrez un bug dans le logiciel, on vous donne l'adresse de la personne à contacter pour le rapporter.
- `COPYRIGHT` : le *copyright*, c'est-à-dire la licence d'utilisation de la commande. La plupart des programmes que vous utilisez sont certainement des programmes open source sous licence GPL, ce qui vous donne le droit de voir la source et de redistribuer le programme librement.
- `SEE ALSO` : cette section vous propose de « voir aussi » d'autres commandes en rapport avec celle que vous êtes en train de regarder. C'est une section parfois intéressante.

Pour faire simple, les trois premières sections sont vraiment les plus importantes (`NAME`, `SYNOPSIS` et `DESCRIPTION`). Ce sont celles que nous regarderons dans la plupart des cas et c'est aussi pour cela qu'elles sont au début du manuel. ;-)

La langue des pages du manuel



Eh mais… c'est tout en anglais ! Tout le monde n'est pas un crack en anglais… comment on fait, nous, si on a du mal à lire l'anglais ?

Vous n'allez peut-être pas apprécier, mais je tiens à vous donner un conseil, le meilleur : familiarisez-vous avec l'anglais. Si vous voulez travailler dans l'informatique, c'est de toute façon une langue incontournable ; il est impossible de l'ignorer ou ce serait alors du pur suicide.

Bon. Malgré tout, il y en a peut-être parmi vous qui ne comptent pas forcément travailler dans l'informatique mais qui aimeraient éviter d'avoir à apprendre l'anglais pour se servir de Linux, ce que je peux très bien comprendre. Si l'évocation du mot « anglais » provoque chez vous des éruptions cutanées inexplicables, alors voici une bonne nouvelle : il existe une version française des pages de manuel !

Vous pouvez l'installer grâce à la commande `apt-get` qu'on a justement apprise dans le chapitre précédent comme par hasard

(rien n'est laissé au hasard, je vous l'ai dit.) :

Code : Console

```
apt-get install manpages-fr
```

Le paquet à installer est `manpages-fr` comme vous pouvez le voir.

Maintenant, si vous tapez `man mkdir`, vous verrez la page s'afficher en français (figure suivante).

```
Terminal - mateo21@simple-it-pc1: ~
Fichier  Éditer  Affichage  Terminal  Aller  Aide
MKDIR(1)                          Commandes                          MKDIR(1)

NOM
       mkdir - Créer des répertoires

SYNOPSIS
       mkdir [OPTION] RÉPERTOIRE...

DESCRIPTION
       Créer les RÉPERTOIREs s'il n'existent pas.

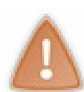
       -Z,  --context=CONTEXTE  (SELinux)  imposer  CONTEXTE comme contexte de
              sécurité

       Les paramètres obligatoires pour les options de forme  longue  le  sont
       aussi pour les options de forme courte.

       -m, --mode=MODE
              définir MODE comme droit d'accès (comme dans chmod), mais pas au
              format umask (rwxrwxrwx)

       -p, --parents
              créer des répertoires  parent  (répertoires  intermédiaires)  si
 Manual page mkdir(1) line 1
```

Pages du manuel en français



Utiliser les pages du manuel en français **n'est pas recommandé**. En effet, ces pages ne sont parfois pas à jour, certaines ne sont pas traduites, et seule la version anglaise est assurée d'être la plus à jour et de comporter le moins d'erreurs. D'ailleurs, en parlant d'erreur cher lecteur, une faute d'orthographe s'est glissée dans la capture d'écran de la figure ci-dessus, sauras-tu la retrouver ?

Dans la suite de ce livre, je considérerai que vous travaillez avec les pages du manuel **en anglais**. C'est vraiment ce que je recommande. Utilisez la version française uniquement si vous ne comprenez rien à l'anglais, sinon faites un petit effort ; je vous assure que ça vaut le coup.

Je vais donc désinstaller les pages du manuel en français et revenir à la version anglaise pour la suite de ce chapitre :

Code : Console

```
apt-get autoremove manpages-fr
```

Bien, les présentations étant faites, passons à la suite !
Nous allons apprendre à lire la section `SYNOPSIS`, une des sections les plus importantes.

Comprendre le SYNOPSIS

Le `SYNOPSIS` est une des sections les plus importantes mais aussi une des plus difficiles à lire.



Quel est le rôle du `SYNOPSIS` ?

Son rôle est de **lister toutes les façons possibles d'utiliser la commande**. En clair, le `SYNOPSIS` vous affiche toutes les combinaisons de paramètres que l'on peut réaliser avec cette commande.
Certains `SYNOPSIS` sont simples, d'autres plus compliqués. Je pense que le mieux est de voir des exemples pour bien comprendre comment ça fonctionne.

man mkdir

Commençons par le man de `mkdir`. La section d'introduction du manuel nous dit « `mkdir - make directories` », ce qui signifie que `mkdir` sert à créer des répertoires.

SYNOPSIS

La section `SYNOPSIS` de `mkdir` est présentée sur la figure suivante.

```
mkdir [OPTION] DIRECTORY...
```

SYNOPSIS de mkdir

Même si ce `SYNOPSIS` est court, il contient déjà beaucoup d'informations : il vous dit comment on doit utiliser la commande. Détaillons point par point ce `SYNOPSIS`.

- `mkdir` : pour utiliser la commande `mkdir`, vous devez commencer par taper `mkdir` ; ça, c'est logique.
- `[OPTION]` : après `mkdir`, vous pouvez écrire une option. Dans le `SYNOPSIS`, on met des crochets pour indiquer que c'est facultatif. Vous n'êtes donc pas obligés d'écrire une option.
- `DIRECTORY` : c'est le nom du répertoire à créer. Ce paramètre est obligatoire puisqu'il n'est pas entre crochets. C'est en effet logique : la commande `mkdir` sert à créer un dossier, la moindre des choses est d'indiquer le nom du dossier à créer !
- ... : le terme `DIRECTORY` est suivi de points de suspension. Cela signifie que l'on peut répéter `DIRECTORY` autant de fois que l'on veut. Traduction : on peut indiquer plusieurs répertoires à la fois pour que la commande les crée tous d'un coup.



Pourquoi **mkdir** est écrit en gras tandis que `OPTION` et `DIRECTORY` sont soulignés ?

Les mots du `SYNOPSIS` écrits en gras sont des mots à taper tels quels. Les mots soulignés, eux, doivent être remplacés par le nom approprié.

C'est logique : on doit bel et bien écrire précisément `mkdir`, par contre on ne doit pas écrire `DIRECTORY` mais le nom du répertoire. La présence d'un paramètre souligné signifie donc : « Remplacez le mot souligné par un mot qui convient à votre cas » .

Exemples d'utilisation

D'après le `SYNOPSIS`, on doit au minimum écrire le nom du dossier. Par exemple :

Code : Console

```
mkdir images
```

Comme on l'a vu, les points de suspension après `DIRECTORY` nous indiquent qu'on peut répéter le nom du répertoire autant de

fois que l'on veut, ce qui nous permet d'en créer plusieurs d'un coup.
On peut donc aussi utiliser `mkdir` comme ceci :

Code : Console

```
mkdir images videos musiques
```

… ce qui aura pour effet de créer trois dossiers : `images`, `videos` et `musiques`.

Maintenant, on peut aussi préciser des options facultatives. Ces options sont listées dans la section `DESCRIPTION` du man juste un peu plus bas :

Code : Console

```
DESCRIPTION
       Create the DIRECTORY(ies), if they do not already exist.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -m, --mode=MODE
              set file mode (as in chmod), not a=rwx - umask

       -p, --parents
              no error if existing, make parent directories as needed

       -v, --verbose
              print a message for each created directory

       -Z, --context=CTX
              set the SELinux security context of each  created  directory  to
              CTX

       --help display this help and exit

       --version
              output version information and exit
```

Toutes ces options peuvent être utilisées à la place de `[OPTION]` dans le `SYNOPSIS`.
Par exemple, l'option `-v` (ou `--verbose`), c'est pareil mais plus long) affiche un message après chaque répertoire créé. On peut donc écrire :

Code : Console

```
mkdir -v images videos musiques
```

Résultat :

Code : Console

```
mateo21@mateo21-desktop:~/tests$ mkdir -v images videos musiques
mkdir: création du répertoire `images'
mkdir: création du répertoire `videos'
mkdir: création du répertoire `musiques'
```

La commande nous informe maintenant de ce qu'elle fait. Sans le `-v`, la commande n'affiche rien (on dit qu'elle est silencieuse).

Vous remarquerez d'ailleurs qu'on retrouve l'option `-v` dans beaucoup de commandes. Elle a chaque fois la même signification : elle demande à la commande d'afficher le détail de ce qu'elle est en train de faire. On dit alors qu'on utilise la commande en mode « verbeux » (bavard), pour bien voir tout ce qu'elle fait.

man cp

Essayons une commande un peu plus complexe : `cp`. Je vous rappelle que cette commande sert à copier des fichiers et des répertoires.

SYNOPSIS

Son `SYNOPSIS` est présenté sur la figure suivante.

```
cp [OPTION]... [-T] SOURCE DEST
cp [OPTION]... SOURCE... DIRECTORY
cp [OPTION]... -t DIRECTORY SOURCE...
```

SYNOPSIS de cp

Là, ça commence à devenir un peu plus délicat.

Pourquoi y a-t-il trois lignes, déjà ? Tout simplement parce qu'on ne pouvait pas exprimer toutes les façons d'utiliser `cp` sur une seule ligne.

- Commençons par la première ligne :
  `cp [OPTION]… [-T] SOURCE DEST`La seule chose obligatoire ici ce sont les paramètres `SOURCE` (le nom du fichier à copier) et `DEST` (le nom de la copie à créer).Ces fichiers peuvent être précédés d'une ou plusieurs options (remarquez les points de suspension) ainsi que de l'option `-T`.
- La seconde ligne est un peu différente :
  `cp [OPTION]… SOURCE… DIRECTORY`Cette fois, on peut copier un ou plusieurs fichiers (`SOURCE…`) vers un répertoire (`DIRECTORY`). Tout cela peut encore une fois être précédé d'une ou plusieurs options.
- Enfin, la troisième ligne :
  `cp [OPTION]… -t DIRECTORY SOURCE…`… signifie qu'on peut aussi écrire le répertoire (`DIRECTORY`) dans un premier temps, suivi d'un ou plusieurs fichiers (`SOURCE…`). Attention, vous remarquez que dans ce cas il est obligatoire d'utiliser le paramètre `-t` qui n'est plus entre crochets.

Exemples d'utilisation

Ça fait beaucoup de façons d'utiliser `cp`, en fait.

Si on se base sur la première ligne, on peut juste écrire :

Code : Console

```
cp photo.jpg photo_copie.jpg
```

… ce qui aura pour effet de créer la copie `photo_copie.jpg`.

On peut aussi ajouter une ou plusieurs options. Pour connaître toutes les options disponibles, vous devrez lire la section `DESCRIPTION`. Pour `cp`, il y a beaucoup de choix comme vous pouvez le voir.
Par exemple, on retrouve notre mode `-v` (verbeux) qui demande à la commande de détailler ce qu'elle fait. On pourrait aussi ajouter `-i` qui demande confirmation si le fichier de destination existe déjà.

On peut donc faire :

Code : Console

```
cp -vi photo.jpg photo_copie.jpg
```

Dans mon cas, le fichier `photo_copie.jpg` existait déjà. L'ajout de l'option `-i` va me demander confirmation pour savoir si je veux bien écraser le fichier. Je peux répondre par « o » ou « n » (pour oui ou non), ou « y » ou « n » (pour *yes* ou *no*).

Code : Console

```
mateo21@mateo21-desktop:~$ cp -vi photo.jpg photo_copie.jpg
cp: écraser `photo_copie.jpg'? o
`photo.jpg' -> `photo_copie.jpg'
```

Comme le fichier existait déjà, on m'a demandé confirmation. La dernière ligne est le résultat du mode verbeux qu'on a demandé.

Bien. Tout ça c'était juste pour la première ligne, dans le cas où l'on veut copier un fichier. Essayons un peu ce que propose la seconde ligne : copier un ou plusieurs fichiers dans un dossier.

Code : Console

```
cp photo.jpg photo_copie.jpg images/
```

Là, on exploite la seconde façon d'utiliser `cp` (seconde ligne du `SYNOPSIS`). On copie deux fichiers dans le sous-dossier `images/`.
Bien entendu, comme l'indique le `SYNOPSIS`, on peut là encore utiliser des options, comme `-v` et `-i` que l'on vient de voir.



Mon conseil : vous vous demandez peut-être comment je comprends la signification de chaque mot du `SYNOPSIS`. Par exemple, était-il évident de savoir que `SOURCE` correspondait au fichier que l'on voulait copier ?
Si je sais tout ça, c'est parce que j'ai lu attentivement le début de la section `DESCRIPTION` du man. Je vous recommande d'en faire de même quelle que soit la commande que vous êtes en train d'analyser.

Code : Console

```
DESCRIPTION
       Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.
```



Cette phrase est claire, pour peu que l'on comprenne l'anglais.
Elle signifie que l'on peut copier le paramètre appelé `SOURCE` vers `DEST`, ou encore (cas de la seconde ligne) plusieurs fichiers `SOURCE` vers un répertoire (`DIRECTORY`).

man apt-get

Plus joyeux encore : le `SYNOPSIS` de la commande `apt-get` (figure suivante).

```
apt-get [-hvs] [-o=option de configuration] [-c=fichier] {[update] |
        [upgrade] | [dselect-upgrade] | [install paquet...] |
        [remove paquet...] | [source paquet...] | [build-dep paquet...]
        | [check] | [clean] | [autoclean]}
```

SYNOPSIS de apt-get

La bonne nouvelle cette fois c'est que les créateurs d'`apt-get` n'ont pas eu besoin de faire tenir la commande sur trois lignes : ils ont mis tous les cas d'utilisation possibles sur une seule ligne !
La mauvaise nouvelle, c'est que c'est un peu difficile à lire comme ça.

Décortiquons.

La commande `apt-get` doit commencer par « apt-get » (ce mot est d'ailleurs écrit en gras). Ça, c'est logique.

Ensuite, vous pouvez utiliser une des options `-hvs` (vous pouvez utiliser juste `-h`, mais aussi `-hv`, `-v`, `-vs`, `-hvs`…). Ces options étant entre crochets, elles sont facultatives.

Pareil ensuite pour `-o` et `-c` ; ces options sont facultatives. En revanche, vous remarquerez qu'elles doivent être obligatoirement suivies d'une valeur. Par exemple `-o=option de configuration`. Je vous rappelle que le fait que `option de configuration` soit souligné signifie que vous ne devez pas recopier ces mots tels quels dans la console : vous devez les remplacer par une valeur qui convient (lisez la section `DESCRIPTION` pour en savoir plus sur `-o`).

La section qui m'intéresse et que je voudrais qu'on analyse plus en détail arrive juste après. Elle commence et se termine par des accolades :

Code : Console

```
{[update] | [upgrade] | [dselect-
upgrade] | [install paquet…] | [remove paquet…] | [source paquet…] | [build-
dep paquet…] | [check] | [clean] | [autoclean]}
```

Vous remarquerez qu'à l'intérieur les mots sont séparés par des barres verticales « `|` ». Ces barres verticales signifient « OU » , ce qui veut dire que vous devez mettre une et une seule option issue de la liste entre accolades.
Parmi ces options possibles, il y en a que vous devez connaître maintenant, comme :

- `update` : met à jour le cache des paquets disponibles sur votre ordinateur ;
- `upgrade` : met à jour tous les paquets installés si une nouvelle version est disponible ;
- `install paquet…` : installe le ou les paquets demandés. La présence des points de suspension après « paquet » signifie que vous pouvez indiquer plusieurs paquets à installer d'un coup ;
- etc.

Il y a bien d'autres mots clés utilisables. Pour voir la signification de chacun d'eux, je vous invite à lire la section `DESCRIPTION` du man qui sert précisément à expliquer cela.

Exemples d'utilisation

Le `SYNOPSIS` indique donc qu'on doit choisir une des options entre accolades séparées par des barres verticales.
On peut donc écrire :

Code : Console

```
apt-get install monpaquet
```

Ou encore :

Code : Console

```
apt-get update
```

Ou encore :

Code : Console

```
apt-get autoclean
```

En revanche, **on ne peut pas utiliser simultanément deux options séparées par une barre verticale** :

Code : Console

```
INTERDIT :
apt-get update install monpaquet
```

Le `SYNOPSIS` nous avait bien dit : « Utilisez `update` OU `install` OU `upgrade` (OU ...), mais pas deux éléments de cette liste à la fois ».
Souvenez-vous donc que les barres verticales signifient « OU » et tout ira bien.

Résumé de la syntaxe du `SYNOPSIS`

Voici un petit résumé de la syntaxe du `SYNOPSIS` pour vous souvenir de la façon dont chaque élément doit être interprété :

- **gras** : tapez le mot exactement comme indiqué ;
- `souligne` : remplacez le mot souligné par la valeur qui convient dans votre cas ;
- `[-hvc]` : toutes les options `-h`, `-v` et `-c` sont facultatives ;
- `a|b` : vous pouvez écrire l'option « a » OU « b », mais pas les deux à la fois ;
- `option...` : les points de suspension indiquent que l'option peut être répétée autant de fois que vous voulez.

apropos : trouver une commande

Le man suppose que vous connaissez déjà votre commande et que vous voulez en savoir plus. Mais si vous ne connaissez pas la commande, comment faites-vous ?

C'est là que la commande `apropos` intervient. Vous lui donnez en paramètre un mot clé et elle va le rechercher dans les descriptions de toutes les pages du manuel.
La commande `apropos` est donc un peu l'inverse de `man` : elle vous permet de retrouver une commande.

Prenons un exemple : vous recherchez une commande (que vous avez installée) en rapport avec le son parce que vous aimeriez bien savoir comment modifier le volume en console.
Vous pouvez taper :

Code : Console

```
apropos sound
```

... ce qui va rechercher toutes les commandes qui parlent de son (`sound`) dans leur page du manuel.

Résultat :

Code : Console

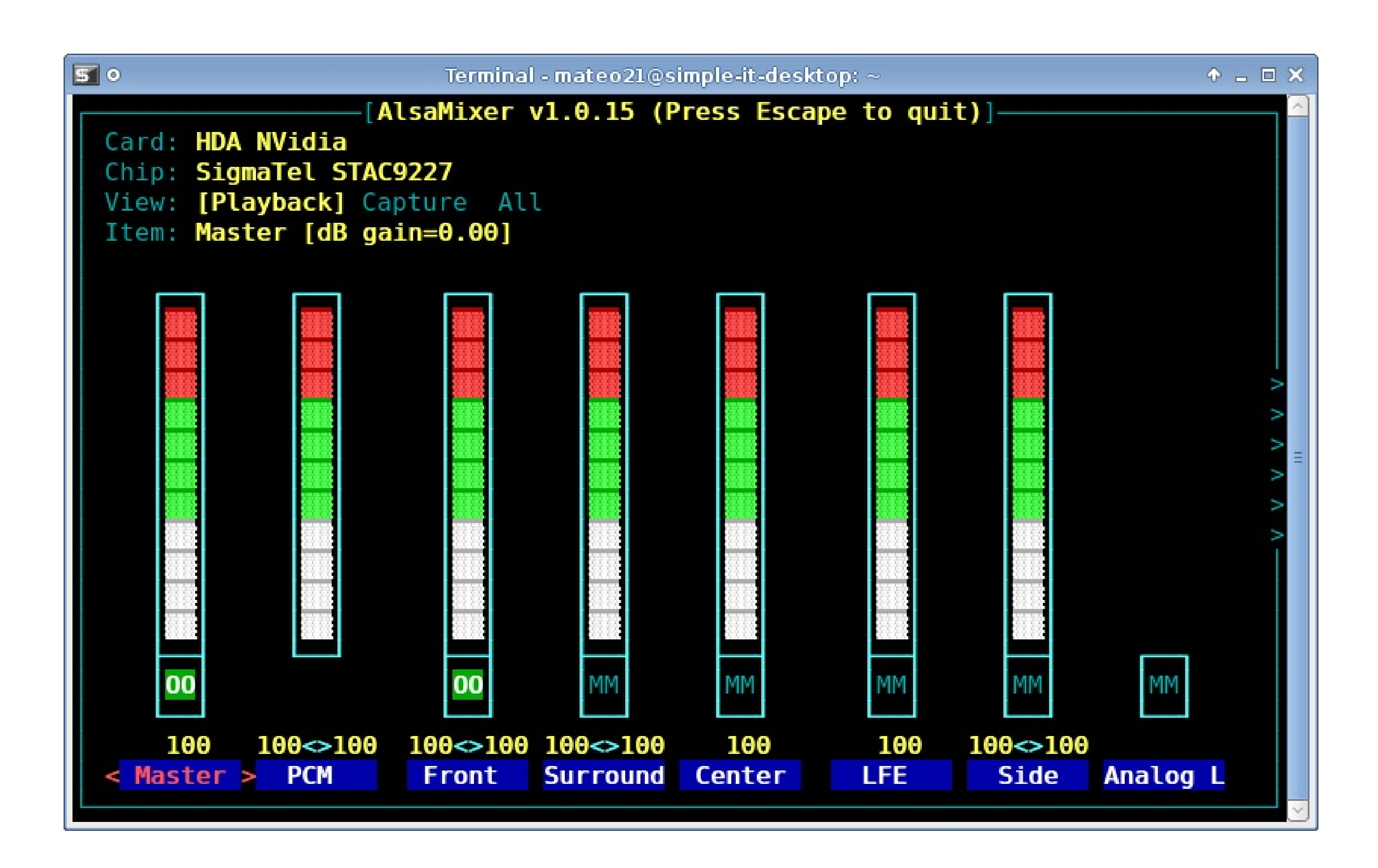
```
mateo21@mateo21-desktop:~$ apropos sound
alsactl (1)          - advanced controls for ALSA soundcard driver
alsamixer (1)        -
 soundcard mixer for ALSA soundcard driver, with ncurse...
amixer (1)           - command-line mixer for ALSA soundcard driver
aplay (1)            - command-
line sound recorder and player for ALSA soundc...
arecord (1)          - command-
```

```
line sound recorder and player for ALSA soundc...
artscat (1)          - pipe data to sound device
asoundconf (1)       -
 utility to read and change the user's ALSA library con...
```

À gauche la commande, à droite l'extrait de sa courte description dans laquelle `apropos` a trouvé le mot que vous recherchiez. Il se trouve que ce que je cherchais était `alsamixer` (figure suivante). Et zou ! :-)

Code : Console

```
alsamixer
```



Et si vous voulez en savoir plus sur la commande `alsamixer`, vous savez maintenant comment faire !

Code : Console

```
man alsamixer
```

```
Terminal - mateo21@simple-it-desktop: ~
ALSAMIXER(1)                                                   ALSAMIXER(1)

NAME
       alsamixer  -  soundcard  mixer  for ALSA soundcard driver, with ncurses
       interface

SYNOPSIS
       alsamixer [options]

DESCRIPTION
       alsamixer is an ncurses mixer program for use with the  ALSA  soundcard
       drivers. It supports multiple soundcards with multiple devices.

OPTIONS
       -h, -help
              Help: show available flags.

       -c <card number or identification>
              Select  the  soundcard  to use, if you have more than one. Cards
              are numbered from 0 (the default).

       -D <device identification>
              Select the mixer device to control.
 Manual page alsamixer(1) line 1
```

Son `SYNOPSIS`, présenté sur la figure suivante, est ridiculement simple. Pfeuh ! Même pas drôle.

D'autres façons de lire le manuel

Bien que ce soit la technique la plus courante, utiliser `man` et `apropos` n'est pas le seul moyen de vous documenter. Quelles sont les alternatives à `man` ?

Le paramètre -h (et --help)

Bien que ça ne soit pas une règle, la plupart des commandes acceptent un paramètre `-h` (et parfois son équivalent plus long `--help`) qui provoque l'affichage d'une aide résumée. Parfois cette aide est d'ailleurs plus facile à lire que celle du man, ce qui fait qu'il m'arrive de l'utiliser de temps en temps.

Par exemple :

Code : Console

```
apt-get -h
```

Ce qui nous donne :

Code : Console

```
mateo21@mateo21-desktop:~$ apt-get -h
apt 0.7.9ubuntu15 pour amd64 compilé sur Mar 14 2008 00:00:28
Usage: apt-get [options] command
       apt-get [options] install|remove pkg1 [pkg2 ...]
       apt-get [options] source pkg1 [pkg2 ...]

apt-get is a simple command line interface for downloading and
installing packages. The most frequently used commands are update
and install.
```

```
Commands:
   update - Retrieve new lists of packages
   upgrade - Perform an upgrade
   install - Install new packages (pkg is libc6 not libc6.deb)
   remove - Remove packages
   autoremove - Remove all automatic unused packages
   purge - Remove and purge packages
   source - Download source archives
   build-dep - Configure build-dependencies for source packages
   dist-upgrade - Distribution upgrade, see apt-get(8)
   dselect-upgrade - Follow dselect selections
   clean - Erase downloaded archive files
   autoclean - Erase old downloaded archive files
   check - Verify that there are no broken dependencies

Options:
  -h  This help text.
  -q  Loggable output - no progress indicator
  -qq No output except for errors
  -d  Download only - do NOT install or unpack archives
  -s  No-act. Perform ordering simulation
  -y  Assume Yes to all queries and do not prompt
  -f  Attempt to correct a system with broken dependencies in place
  -m  Attempt to continue if archives are unlocatable
  -u  Show a list of upgraded packages as well
  -b  Build the source package after fetching it
  -V  Show verbose version numbers
  -c=? Read this configuration file
  -o=? Set an arbitrary configuration option, eg -o dir::cache=/tmp
See the apt-get(8), sources.list(5) and apt.conf(5) manual
pages for more information and options.
                       This APT has Super Cow Powers.
```

Le `-h` est parfois un bon complément au man si vous n'arrivez pas à comprendre comment utiliser la commande. On y trouve parfois des informations utiles comme ici : « *The most frequently used commands are update and install* », ce qui signifie que l'on utilise le plus souvent `apt-get` avec les paramètres `update` et `install` (et c'est vrai).



Parfois, il n'y a pas de page de manuel pour une commande (`man` ne fonctionne pas pour cette dernière) mais le `-h` ou le `--help` fonctionne. Pensez-y !

La commande whatis

La commande `whatis` est une sorte de `man` très allégé. Elle donne juste l'en-tête du manuel pour expliquer en deux mots à quoi sert la commande. Par exemple pour `mkdir` :

Code : Console

```
whatis mkdir
```

Ça vous permet d'éviter de sortir l'artillerie lourde juste pour savoir à quoi sert la commande.

Code : Console

```
mateo21@mateo21-desktop:~$ whatis mkdir
mkdir (1)            - make directories
```

Rechercher `man` sur le Web

Enfin, il est bien de le préciser : on retrouve aussi le man sur le Web !
Si vous devez lire un manuel et que vous n'êtes pas sous Linux à ce moment-là, utilisez tout bêtement un moteur de recherche comme Google pour retrouver la doc.

Par exemple, vous pouvez taper la recherche : `man mkdir`.
Je vous parie que vous trouverez le manuel dans les premiers liens qui s'offrent à vous.

Bonne recherche !

En résumé

- Sous Linux, toutes les commandes et leurs paramètres sont documentés dans le manuel. Il est recommandé de lire le manuel à chaque fois que vous avez des questions sur une commande car la réponse s'y trouve la plupart du temps.
- On fait appel au manuel avec la commande `man` suivie du nom de la commande sur laquelle on veut avoir plus d'informations. Par exemple : `man mkdir`.
- Dans le manuel, on se déplace avec les touches fléchées ou `Page Up` et `Page Down`, on fait une recherche avec la touche `/` (slash) et on quitte avec la touche `Q`.
- Le manuel d'une commande commence toujours par son `SYNOPSIS` : c'est un résumé des différentes manières d'utiliser la commande. Les options facultatives sont écrites entre crochets.
- Pour trouver une commande correspondant à un certain usage, utilisez `apropos`. Ainsi, `apropos sound` affichera toutes les commandes ayant un rapport avec le son.

 Rechercher des fichiers

Sous Linux, les fichiers sont organisés d'une façon assez particulière. Nous l'avons vu en affichant la liste des répertoires à la racine avec un `ls /`, il y a une foule de dossiers aux noms assez variés : `var`, `opt`, `etc`, `bin`, `dev`…

Une partie de ces répertoires est là pour des raisons historiques, depuis l'époque d'Unix. Le problème, c'est qu'il peut être difficile de retrouver le fichier dont on a besoin dans cette foule de répertoires.

Pas de panique ! On dispose heureusement sous Linux d'outils très puissants pour rechercher un fichier sur le disque dur. Certains d'entre eux sont très rapides, d'autres plus lents mais aussi plus complets.

Partons à la recherche de ces fichiers !

locate : une recherche rapide

La première façon d'effectuer une recherche que nous allons voir est de loin la plus simple. La commande s'appelle `locate` (« localiser »). Elle est très rapide.

Utiliser `locate`

Son utilisation est intuitive, il suffit d'indiquer le nom du fichier que vous voulez retrouver. Par exemple :

Code : Console

```
mateo21@mateo21-desktop:~$ locate notes.txt
/home/mateo21/notes.txt
```

La commande a retrouvé notre fichier `notes.txt` qui était situé dans `/home/mateo21`.

Essayons maintenant de retrouver ces vieilles photos d'Australie…

Code : Console

```
mateo21@mateo21-desktop:/var/log$ locate australie
/home/mateo21/photos/australie1.jpg
/home/mateo21/photos/australie2.jpg
/home/mateo21/photos/australie3.jpg
```

`locate` vous donne tous les fichiers qui contiennent le mot « australie » dans leur nom. Que ce soient des fichiers ou des dossiers, elle ne fait pas la différence. Elle vous donne la liste complète des fichiers qu'elle a trouvés.



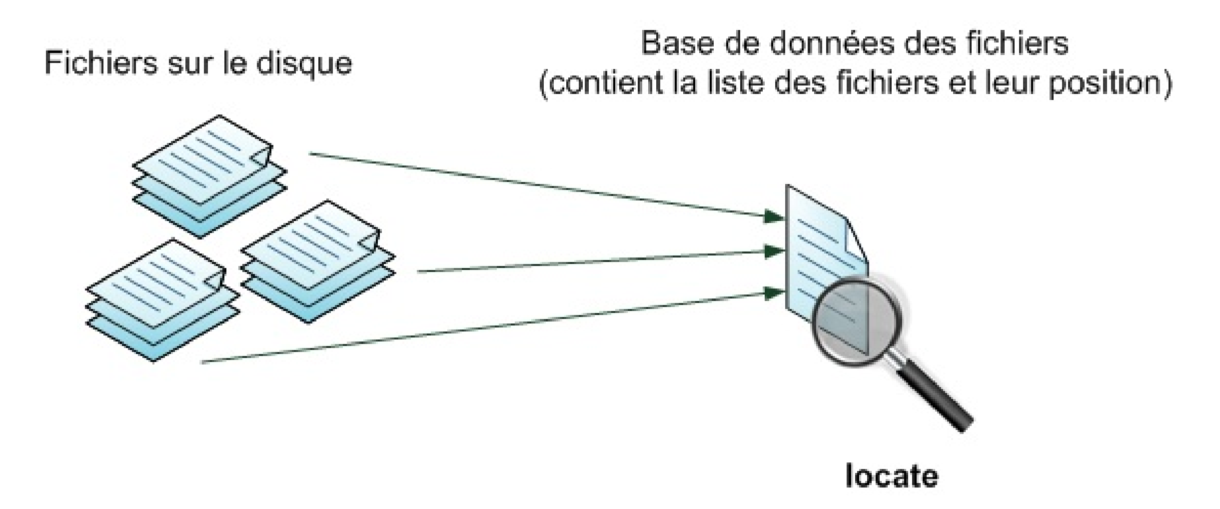
Il existe aussi la commande `slocate` qui est un peu plus récente, mais on ne la retrouve pas sur toutes les distributions par défaut. Vous pouvez toutefois l'installer rapidement avec un `apt-get` si vous ne l'avez pas.
La commande `slocate` est identique à `locate`, à la différence près qu'elle vérifie les droits des fichiers avant de les lister. Avec `slocate`, un utilisateur ne pourra donc pas voir un nom de fichier apparaître s'il n'a pas le droit de le lire, tandis qu'avec un `locate` le nom du fichier serait apparu dans les résultats quand même.

La base de données des fichiers



Je ne comprends pas. Je viens de créer des fichiers (avec la commande `touch` par exemple), et `locate` ne me renvoie aucun résultat. Pourquoi ?

C'est justement le défaut de `locate` dont je voulais vous parler : la commande ne fait pas la recherche sur votre disque dur entier, mais seulement sur une base de données de vos fichiers (figure suivante).



Votre problème, c'est que les fichiers viennent tout juste d'être créés et n'ont pas encore été répertoriés dans la base de données. Ils ne seront donc pas découverts par `locate`.

Une fois par jour, votre système mettra à jour la base de données. Donc, si vous réessayez demain, il est probable que `locate` trouve enfin votre fichier.


Mais… je ne vais pas attendre 24 h pour retrouver un fichier, tout de même !

Non, bien sûr !

Vous pouvez forcer la commande `locate` à reconstruire la base de données des fichiers du disque dur. Cela se fait avec la commande `updatedb`, à exécuter en root (avec `sudo`) :

Code : Console

```
sudo updatedb
```

La mise à jour de la liste des fichiers peut être un peu longue, il faudra patienter. Une fois que c'est fini, réessayez de faire un `locate`, il devrait maintenant trouver votre fichier.

En résumé, `locate` est pratique car rapide et facile à utiliser.
Cependant, `locate` donne parfois trop de résultats car elle recherche dans tous les répertoires du disque dur, elle n'est donc pas très précise. De plus, les fichiers qui viennent tout juste d'être créés ne seront pas découverts, à moins d'exécuter `updatedb`.

Quand `locate` ne suffit pas, on a besoin d'une commande plus puissante. On sort l'artillerie lourde : `find`.

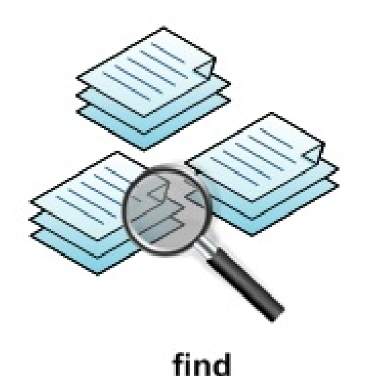
find : une recherche approfondie

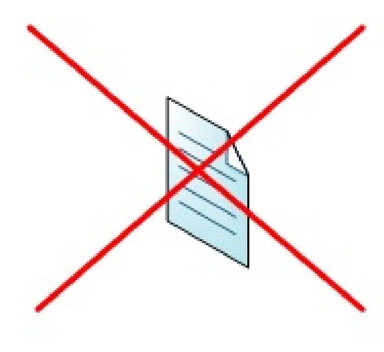
`find` est la commande de recherche par excellence pour retrouver des fichiers, mais aussi pour effectuer des opérations sur chacun des fichiers trouvés. Elle est très puissante, permet donc de faire beaucoup de choses, et par conséquent… elle est un peu complexe.

Savoir la manipuler est toutefois indispensable, donc découvrons-la !

`find` recherche les fichiers actuellement présents

Contrairement à `locate`, `find` ne va pas lire dans une base de données mais au contraire parcourir tout votre disque dur (figure suivante). **Cela peut être très long** si vous avez plusieurs giga-octets de données !





Avec cette méthode de recherche, vous ne risquez pas de rater un fichier récent qui aurait pu être créé aujourd'hui. Et ce n'est pas le seul avantage, loin de là !

Fonctionnement de la commande `find`

La commande `find` s'utilise de la façon suivante :

`find` « où » « quoi » « que faire avec »

Seul le paramètre « quoi » est obligatoire.

- **Où** : c'est le nom du dossier dans lequel la commande va faire la recherche. Tous les sous-dossiers seront analysés. Contrairement à `locate`, il est donc possible de limiter la recherche à `/home` par exemple.
  Par défaut, si ce paramètre n'est pas précisé, la recherche s'effectuera dans le dossier courant et ses sous-dossiers.
- **Quoi** : c'est le fichier à rechercher. On peut rechercher un fichier par son nom, mais aussi en fonction de la date de sa création, de sa taille, etc.
  Ce paramètre est obligatoire.
- **Que faire avec** : il est possible d'effectuer des actions automatiquement sur chacun des fichiers trouvés (on parle de « post-traitement »). L'action la plus courante consiste à afficher simplement la liste des fichiers trouvés, mais nous verrons que nous pouvons faire bien d'autres choses.
  Par défaut, la commande `find` affiche les résultats trouvés et ne fait rien d'autre avec.

Utilisation basique de la commande `find`

Commençons doucement et voyons des exemples pratiques. Nous allons tout d'abord rechercher un fichier et afficher sa position.

Recherche à partir du nom

Je vais utiliser dans un premier temps le seul paramètre obligatoire : le nom du fichier à retrouver.

Je me place dans mon répertoire home et je vais essayer de retrouver un fichier appelé `logo.png` que j'ai égaré. Je dois écrire :

Code : Console

```
find -name "logo.png"
```

Le `-name "logo.png"` est un paramètre qui demande de retrouver un fichier qui s'appelle très exactement `logo.png`.

Voici le résultat :

Code : Console

```
mateo21@mateo21-desktop:~$ find -name "logo.png"
/home/mateo21/projet/images/logo.png
```

Si la recherche n'affiche rien, cela signifie qu'aucun fichier n'a été trouvé.



Comme nous n'avons pas précisé où rechercher, `find` a cherché dans le répertoire dans lequel nous nous trouvions et dans ses sous-répertoires. Le ~
de l'invite de commandes signifie que j'étais dans mon home, c'est-à-dire dans `/home/mateo21/`. Tous les sous-répertoires ont été analysés.}

Maintenant, si je suis dans mon home mais que je veux rechercher dans un autre répertoire, il faudra préciser en premier paramètre le répertoire dans lequel chercher.
Par exemple, si je veux retrouver tous les fichiers qui s'appellent `syslog` situés dans `/var/log` (et ses sous-répertoires), je dois écrire :

Code : Console

```
find /var/log/ -name "syslog"
```

Essayons ça !

Code : Console

```
mateo21@mateo21-desktop:~$ find /var/log/ -name "syslog"
/var/log/syslog
/var/log/installer/syslog
```

Les paramètres correspondent à ceux que je vous ai présentés un peu plus tôt :

- **où** : dans `/var/log` (et ses sous-dossiers) ;
- **quoi** : le fichier `syslog`.

Il n'y a pas de paramètre « que faire avec », donc par défaut `find` choisit d'afficher les fichiers trouvés.

Notez que, contrairement à `locate`, `find` récupère uniquement la liste des fichiers qui s'appellent exactement comme demandé. Ainsi, s'il existe un fichier nommé `syslog2`, il ne sera pas listé dans les résultats. Pour qu'il le soit, il faut utiliser le joker : l'étoile « * » !

Exemple :

Code : Console

```
mateo21@mateo21-desktop:~$ find /var/log/ -name "syslog*"
/var/log/syslog.3.gz
/var/log/syslog.5.gz
/var/log/syslog.4.gz
/var/log/syslog
/var/log/syslog.6.gz
/var/log/syslog.2.gz
/var/log/syslog.1.gz
/var/log/installer/syslog
/var/log/syslog.0
```

Ici, nous avons affiché tous les fichiers qui commençaient par « syslog ».

Si on avait voulu avoir la liste des fichiers qui se terminent par « syslog », on aurait écrit `"*syslog"`.
De même, si on avait voulu avoir la liste des fichiers qui contiennent « syslog », que ce soit au début, au milieu ou à la fin, on aurait écrit `"*syslog*"`.

L'étoile est donc un joker qui signifie « il peut y avoir n'importe quel caractère à cet endroit ».

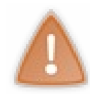


Et si je veux rechercher sur tout le disque dur, et pas seulement dans un dossier ?

Facile, il suffit d'indiquer le répertoire racine `/`. Je vous rappelle que sous Linux, tous les dossiers sont situés dans un sous-niveau de `/`. C'est un petit peu l'équivalent du `C:` sous Windows.

Code : Console

```
find / -name "syslog"
```



La recherche depuis la racine prend beaucoup de temps si vous avez de nombreux fichiers. De plus, si vous n'êtes pas root, vous aurez de nombreux messages d'erreur vous indiquant que vous n'avez pas le droit d'aller dans certains répertoires.
En général, à moins d'être très patient (ou désespéré), on ne fait pas de recherche depuis la racine.

Recherche à partir de la taille

Vous ne connaissez pas le nom du fichier que vous recherchez ? Pas de panique !
Il y a bien d'autres façons de retrouver des fichiers (ou des dossiers, d'ailleurs).

Par exemple, on peut rechercher tous les fichiers qui font plus de 10 Mo.

Code : Console

```
mateo21@mateo21-desktop:/var$ find   -size +10M
/home/mateo21/souvenirs.avi
/home/mateo21/backups/backup_mai.gz
/home/mateo21/backups/backup_juin.gz
```



Rappel : le tilde « ~
» signifie « rechercher dans mon home », en l'occurrence chez moi « `/home/mateo21/` ».}

Au lieu de se baser sur le nom, on se base ici sur la taille (`-size`). Le `+10M` indique que l'on recherche des fichiers de plus de 10 Mo. On peut aussi utiliser `k` pour les Ko, `G` pour les Go, etc.
Vous pouvez aussi utiliser un moins « - » à la place du « + » pour obtenir par exemple les fichiers de moins de 10 Mo. Et si vous enlevez le « + », la commande cherchera des fichiers de 10 Mo exactement (ni plus, ni moins).

Recherche à partir de la date de dernier accès

Vous êtes sûrs d'avoir accédé à votre rapport au format `.odt` il y a moins de 7 jours, mais vous n'arrivez pas à le retrouver ?

Avec `-atime`, vous pouvez indiquer le nombre de jours qui vous séparent du dernier accès à un fichier.

Code : Console

```
mateo21@mateo21-desktop:~$ find -name "*.odt" -atime -7
/home/mateo21/ecriture/resume_infos_juin.odt
```

J'ai combiné ici une recherche par le nom avec une recherche par la date. Si je ne me souvenais même plus de l'extension du fichier, j'aurais dû utiliser seulement `-atime`, mais ça m'aurait probablement donné beaucoup de fichiers (en fait, tous les fichiers modifiés depuis 7 jours dans mon répertoire personnel !).

Là encore, vous pouvez utiliser un « + » à la place… ou même enlever le signe pour rechercher un fichier auquel on aurait accédé il y a exactement 7 jours.

Rechercher uniquement des répertoires ou des fichiers

On peut aussi rechercher uniquement des répertoires ou des fichiers.

Utilisez :

- `-type d` : pour rechercher uniquement des répertoires (*directories*) ;
- `-type f` : pour rechercher uniquement des fichiers (*files*).

Par défaut, `find` cherche des répertoires ET des fichiers. Ainsi, si vous avez un fichier appelé `syslog` et un répertoire appelé `syslog`, les deux résultats seront affichés.

Pour obtenir uniquement les répertoires qui s'appellent `syslog` (et non pas les fichiers), tapez donc :

Code : Console

```
find /var/log -name "syslog" -type d
```

Utilisation avancée avec manipulation des résultats

Pour l'instant, nous n'avons pas indiqué de paramètre « que faire avec » pour effectuer une action sur les résultats trouvés. Par défaut, les noms des fichiers trouvés étaient affichés.

En fait,

Code : Console

```
find   -name "*.jpg"
```

… est équivalent à :

Code : Console

```
find   -name "*.jpg" -print
```

`-print` signifie « afficher les résultats trouvés ».
Si le `-print` n'est pas écrit, la commande comprend toute seule qu'elle doit afficher la liste des fichiers.

On peut cependant remplacer ce `-print` par d'autres paramètres.

Afficher les fichiers de façon formatée

Par défaut, on liste juste les noms des fichiers trouvés. On peut cependant avec l'option `-printf`, qui rappellera à certains le langage C, manipuler un peu ce qui est affiché.

Exemple :

Code : Console

```
mateo21@mateo21-desktop:~$ find . -name "*.jpg" -printf "%p - %u\n"
./photos/australie1.jpg - mateo21
./photos/australie2.jpg - mateo21
./photos/australie3.jpg - mateo21
```

Ici, j'affiche le nom du fichier, un tiret et le nom du propriétaire de ce fichier. Le `\n` permet d'aller à la ligne.

Je vous conseille fortement de lire la doc', à la section « -printf » (faites une recherche). Direction : `man find` ! Vous y trouverez tous les éléments utilisables avec `-printf`, en plus du `%p` et du `%u`.

Supprimer les fichiers trouvés

Un des usages les plus courants de `find`, à part retrouver des fichiers, consiste à les supprimer.

Si je veux faire le ménage dans mon home et par exemple supprimer tous mes fichiers « jpg », je vais écrire ceci :

Code : Console

```
find   -name "*.jpg" -delete
```



Soyez bien sûrs de ce que vous faites ! Il n'y aura pas de confirmation !

Et voilà, toutes les images ont disparu.

Appeler une commande

Avec `-exec`, vous pouvez appeler une commande qui effectuera une action sur chacun des fichiers trouvés.

Imaginons que je souhaite mettre un chmod à 600 pour chacun de mes fichiers « jpg », pour que je sois le seul à pouvoir les lire :

Code : Console

```
find   -name "*.jpg" -exec chmod 600 {} \;
```

La commande n'affiche rien s'il n'y a pas eu d'erreur.



Euh… comment ça marche, ce truc ?

Pour chaque fichier `.jpg` trouvé, on exécute la commande qui suit `-exec` :

- cette commande ne doit PAS être entre guillemets ;

- les accolades `{}` seront remplacées par le nom du fichier ;
- la commande doit finir par un `\;` obligatoirement.

C'est un peu compliqué au premier abord, mais c'est très puissant ! Vous pouvez faire ce que vous voulez avec ça.

Exercice : essayez de regrouper tous les fichiers `.jpg` éparpillés dans votre répertoire home dans un dossier `images`.



Si le fait que la commande ne vous demande pas de confirmation vous ennuie, vous pouvez utiliser `-ok` à la place de `-exec`. Le principe est le même, mais on vous demandera une confirmation pour chaque fichier rencontré. Il faudra répondre par « y » (*yes*) ou « n » (*no*) à chaque fois.

En résumé

- Pour rechercher un fichier sur tout le disque, la commande `locate` est très rapide mais ne trouvera pas les fichiers qui viennent d'être créés dans la journée. On peut mettre à jour la liste des fichiers qu'elle connaît en appelant `updatedb`.
- `find` est une commande plus puissante mais plus lente qui va parcourir votre disque à la recherche de vos fichiers. Elle peut s'utiliser avec trois paramètres, dans l'ordre : où chercher, que chercher et que faire avec.
- On peut rechercher des fichiers en fonction de leur nom (`-name`), de leur taille (`-size`), de leur date de dernier accès (`-atime`)…
- Au lieu d'afficher les fichiers trouvés, on peut automatiquement les supprimer avec `-delete` ou exécuter la commande de son choix sur chacun d'eux avec `-exec`.

Partie 3 : Contrôler les processus et les flux de données



Extraire, trier et filtrer des données

Commevous le savez déjà, la plupart des commandes de Linux sont basées sur le modèle du système d'exploitation Unix. Ce sont les mêmes. Certaines s'utilisent de la même manière depuis les années 60 ! Avantage pour les informaticiens : pas besoin de réapprendre à utiliser les mêmes commandes tous les trois mois.

Mais la question que vous devez vous poser est la suivante : comment se fait-il que la plupart de ces commandes n'aient pas changé depuis si longtemps ? La réponse vient du fait qu'elles n'ont pas eu besoin de changer. En effet, la plupart des commandes que vous découvrez sont très basiques : elles accomplissent une tâche et le font bien, mais pas plus. Ce sont les « briques de base » du système.

Dans ce chapitre, nous allons découvrir une série de commandes basiques qui permettent d'extraire, trier et filtrer des données dans des fichiers. Vous utiliserez certaines d'entre elles (comme `grep`) presque tous les jours !

grep : filtrer des données

La commande `grep` est essentielle. De toutes celles présentées dans ce chapitre, il s'agit probablement de la plus couramment utilisée.

Son rôle est de rechercher un mot dans un fichier et d'afficher les lignes dans lesquelles ce mot a été trouvé. L'avantage de cette commande est qu'elle peut être utilisée de manière très simple ou plus complexe (mais plus précise) selon les besoins en faisant appel aux expressions régulières.



Les expressions régulières constituent un moyen très puissant de rechercher un texte. On les utilise non seulement dans la ligne de commandes Linux, mais aussi dans des éditeurs de texte avancés et dans de nombreux langages de programmation tels que PHP. Vous trouverez d'ailleurs deux chapitres assez complets au sujet des expressions régulières dans le livre *Concevez votre site web avec PHP et MySQL* que j'ai rédigé

.

Nous allons commencer par utiliser `grep` de manière très simple ; nous verrons ensuite comment faire des recherches plus poussées avec les expressions régulières.

Utiliser `grep` simplement

La commande `grep` peut s'utiliser de nombreuses façons différentes. Pour le moment, nous allons suivre le schéma ci-dessous :

Code : Console

```
grep texte nomfichier
```

Le premier paramètre est le texte à rechercher, le second est le nom du fichier dans lequel ce texte doit être recherché.

Essayons par exemple de rechercher le mot « alias » dans notre fichier de configuration `.bashrc`. Rendez-vous dans votre répertoire personnel (en tapant `cd`) et lancez la commande suivante :

Code : Console

```
grep alias .bashrc
```

Cette commande demande de rechercher le mot « alias » dans le fichier `.bashrc` et affiche toutes les lignes dans lesquelles le mot a été trouvé.

Résultat :

Code : Console

```
$ grep alias .bashrc

#  /.bash_aliases, instead of adding them here directly.
#if [ -f  /.bash_aliases ]; then
#     .  /.bash_aliases
# enable color support of ls and also add handy aliases
    alias ls='ls --color=auto'
    #alias dir='ls --color=auto --format=vertical'
    #alias vdir='ls --color=auto --format=long'
# some more ls aliases
alias ll='ls -lArth'
#alias la='ls -A'
#alias l='ls -CF'
```

Pas mal, n'est-ce pas ? Comme vous pouvez le voir, `grep` est davantage un outil de filtre qu'un outil de recherche. Son objectif est de vous afficher uniquement les lignes qui contiennent le mot que vous avez demandé.

Notez qu'il n'est pas nécessaire de mettre des guillemets autour du mot à trouver, sauf si vous recherchez une suite de plusieurs mots séparés par des espaces, comme ceci :

Code : Console

```
grep "Site du Zéro" monfichier
```

`-i` : ne pas tenir compte de la casse (majuscules / minuscules)

Par défaut, `grep` tient compte de la casse : il fait la distinction entre les majuscules et les minuscules. Ainsi, si vous recherchez « alias » et qu'une ligne contient « Alias », `grep` ne la renverra pas.

Pour que `grep` renvoie toutes les lignes qui contiennent « alias », peu importent les majuscules et les minuscules, utilisez l'option `-i` :

Code : Console

```
$ grep -i alias .bashrc

# Alias definitions.
#  /.bash_aliases, instead of adding them here directly.
#if [ -f  /.bash_aliases ]; then
#     .  /.bash_aliases
# enable color support of ls and also add handy aliases
    alias ls='ls --color=auto'
    #alias dir='ls --color=auto --format=vertical'
    #alias vdir='ls --color=auto --format=long'
# some more ls aliases
alias ll='ls -lArth'
#alias la='ls -A'
#alias l='ls -CF'
```

On notera que la première ligne renvoyée n'était pas présente tout à l'heure car le mot « Alias » contenait une majuscule. Avec l'option `-i` on peut désormais la voir.

`-n` : connaître les numéros des lignes

Vous pouvez afficher les numéros des lignes retournées avec `-n` :

Code : Console

```
$ grep -n alias .bashrc

49:#  /.bash_aliases, instead of adding them here directly.
52:#if [ -f  /.bash_aliases ]; then
53:#    .  /.bash_aliases
56:# enable color support of ls and also add handy aliases
59:    alias ls='ls --color=auto'
60:    #alias dir='ls --color=auto --format=vertical'
61:    #alias vdir='ls --color=auto --format=long'
64:# some more ls aliases
65:alias ll='ls -lArth'
66:#alias la='ls -A'
67:#alias l='ls -CF'
```

`-v` : inverser la recherche : ignorer un mot

Si, au contraire, vous voulez connaître toutes les lignes qui **ne contiennent pas** un mot donné, utilisez `-v` :

Code : Console

```
$ grep -v alias .bashrc

#  /.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-
doc)
# for examples

# If not running interactively, don't do anything
[ -z "$PS1" ] && return

# don't put duplicate lines in the history. See bash(1) for more options
export HISTCONTROL=ignoredups
# ... and ignore same sucessive entries.
export HISTCONTROL=ignoreboth

# ... (renvoie beaucoup de lignes, je ne mets pas tout ici)
```

Cette fois, on récupère toutes les lignes du fichier `.bashrc` qui ne contiennent pas le mot « alias ».

`-r` : rechercher dans tous les fichiers et sous-dossiers

Si vous ne savez pas dans quel fichier se trouve le texte que vous recherchez, vous pouvez sortir l'artillerie lourde : l'option `-r` (*recursive*). Cette fois, il faudra indiquer en dernier paramètre le **nom du répertoire** dans lequel la recherche doit être faite (et non pas le nom d'un fichier).

Code : Console

```
grep -r "Site du Zéro" code/
```

… recherchera la chaîne « Site du Zéro » dans tous les fichiers du répertoire `code`, y compris dans les sous-dossiers.



Notez que le « / » à la fin n'est pas obligatoire. Sans cela Linux comprendra tout de même très bien qu'il s'agit d'un répertoire.

Code : Console

```
$ grep -r "Site du Zéro" code/

code/intro.html: Nous vous souhaitons la bienvenue sur le Site du Zéro !
code/tpl/define.tpl: Le Site du Zéro
```

Cette fois, le nom du fichier dans lequel la chaîne a été trouvée s'affiche au début de la ligne.



À noter qu'il existe aussi la commande `rgrep` qui est équivalente à `grep -r`.

Utiliser `grep` avec des expressions régulières

Pour faire des recherches plus poussées – pour ne pas dire des recherches *très poussées* –, vous devez faire appel aux expressions régulières. C'est un ensemble de symboles qui va vous permettre de dire à l'ordinateur très précisément ce que vous recherchez.

Je vous propose dans un premier temps de jeter un oeil au tableau suivante regroupant les principaux caractères spéciaux qu'on utilise dans les expressions régulières.

Caractère spécial	Signification
.	Caractère quelconque
^	Début de ligne
$	Fin de ligne
[ ]	Un des caractères entre les crochets
?	L'élément précédent est optionnel (peut être présent 0 ou 1 fois)
*	L'élément précédent peut être présent 0, 1 ou plusieurs fois
+	L'élément précédent doit être présent 1 ou plusieurs fois
|	Ou
( )	Groupement d'expressions



Help ! Je n'ai rien compris. :-(

C'est normal. Pour bien faire, il faudrait un ou deux chapitres entiers sur les expressions régulières. Je n'ai pas vraiment la place ici pour faire un « minicours » sur les expressions régulières, je vous propose donc de jeter un oeil à ces quelques lignes pour apprendre par l'exemple.

Tout d'abord, il faut savoir qu'on doit utiliser l'option `-E` pour faire comprendre à `grep` que l'on utilise une expression régulière.

Code : Console

```
$ grep -E Alias .bashrc

# Alias definitions.
```



Notez que vous pouvez aussi utiliser la commande `egrep` qui équivaut à écrire `grep -E`.

C'est une expression régulière très simple. Elle demande de rechercher le mot « Alias » (avec un A majuscule). Si le mot est présent dans une ligne, cette dernière est renvoyée.

Bon, jusque-là, rien de nouveau ; ça fonctionnait comme ça avant qu'on utilise les expressions régulières. Essayons de pimenter cela en faisant précéder « Alias » d'un accent circonflexe qui signifie que le mot doit être placé au début de la ligne :

Code : Console

```
$ grep -E ^Alias .bashrc
```

Résultat : `grep` ne renvoie rien. En effet, la ligne de tout à l'heure commençait par un `#` et non pas par « Alias ».

En revanche, on a un résultat si on fait ceci :

Code : Console

```
$ grep -E ^alias .bashrc

alias ll='ls -lArth'
```

Cette fois, la ligne commençait bien par « alias ». De même, on aurait pu utiliser un `$` à la fin pour demander à ce que la ligne se termine par « alias ».

Quelques autres exemples que vous pouvez tester :

Code : Console

```
grep -E [Aa]lias .bashrc
```

… renvoie toutes les lignes qui contiennent « alias » ou « Alias ».

Code : Console

```
grep -E [0-4] .bashrc
```

… renvoie toutes les lignes qui contiennent un nombre compris entre 0 et 4.

Code : Console

```
grep -E [a-zA-Z] .bashrc
```

… renvoie toutes les lignes qui contiennent un caractère alphabétique compris entre « a » et « z » ou entre « A » et « Z ».
Je vous ai fait là une introduction très rapide mais il y aurait beaucoup à dire. Si vous voulez en savoir plus sur les expressions régulières, vous trouverez dans mon livre PHP *Concevez votre site web avec PHP et MySQL* (Livre du Zéro) ou sur le Site du Zéro des explications plus complètes.



Comme vous pourrez le constater, les expressions régulières fonctionnent aussi bien sans le `-E`. Pourquoi ? Normalement, cette option sert à activer la gestion des expressions régulières les plus complexes. Dans la pratique, le manuel nous dit que la version GNU de `grep` (celle que l'on utilise sous Linux) ne fait pas de différence, que l'option soit présente ou non. Les expressions régulières sont toujours activées. En clair, vous aurez besoin du `-E` si un jour vous utilisez `grep` sur une autre machine de type Unix mais en attendant, vous pouvez très bien vous en passer. Le `-E` a été conservé pour des raisons de compatibilité.

sort : trier les lignes

La commande `sort` se révèle bien utile lorsqu'on a besoin de trier le contenu d'un fichier.

Pour nos exemples, je vous propose de créer un nouveau fichier (avec `nano` par exemple) appelé `noms.txt` et d'y placer le texte suivant :

Code : Console

```
François
Marcel
Albert
Jean
Stéphane
patrice
Vincent
jonathan
```

Ensuite, exécutez la commande `sort` sur ce fichier :

Code : Console

```
$ sort noms.txt

Albert
François
Jean
jonathan
Marcel
patrice
Stéphane
Vincent
```

Le contenu du fichier est trié alphabétiquement et le résultat est affiché dans la console.
Vous noterez que `sort` ne fait pas attention à la casse (majuscules / minuscules).

-o : écrire le résultat dans un fichier

Le fichier en lui-même n'a pas été modifié lorsque nous avons lancé la commande. Seul le résultat était affiché dans la console.

Vous pouvez faire en sorte que le fichier soit modifié en précisant un nom de fichier avec l'option `-o` :

Code : Console

```
sort -o noms_tries.txt noms.txt
```

… écrira la liste de noms triés dans `noms_tries.txt`.

-r : trier en ordre inverse

L'option `-r` permet d'inverser le tri :

Code : Console

```
$ sort -r  noms.txt

Vincent
Stéphane
patrice
Marcel
jonathan
Jean
François
Albert
```

-R : trier aléatoirement

Cette option permet de trier aléatoirement les lignes d'un fichier. C'est assez amusant et ça peut se révéler utile dans certains cas :

Code : Console

```
$ sort -R noms.txt

patrice
François
Marcel
jonathan
Jean
Albert
Vincent
Stéphane
```

-n : trier des nombres

Le tri de nombres est un peu particulier. En effet, la commande `sort` ne distingue pas si les caractères sont des nombres et va donc par défaut les trier par ordre alphabétique. Par conséquent, le « mot » 129 précèdera 42 alors que ça devrait être l'inverse !

Prenons un exemple. Créez un nouveau fichier `nombres.txt` et placez-y les nombres suivants :

Code : Console

```
36
16
42
129
27
364
```

Triez-les comme vous avez appris à le faire :

Code : Console

```
$ sort nombres.txt

129
16
27
36
364
42
```

Alphabétiquement, ces nombres sont bien triés. Tout ce qui commence par 1 est en premier, puis vient ce qui commence par 2 et ainsi de suite.
Bien sûr, quand on veut trier des nombres, c'est n'importe quoi.

C'est là que l'option `-n` intervient. Elle permet de trier en considérant le texte comme des nombres. Cette fois, le nombre 42 sera bien placé avant 129 !

Code : Console

```
$ sort -n nombres.txt

16
27
36
42
129
364
```

Magique. ;-)

wc : compter le nombre de lignes

La commande `wc` signifie *word count*. C'est donc a priori un compteur de mots mais en fait, on lui trouve plusieurs autres utilités : compter le nombre de lignes (très fréquent) et compter le nombre de caractères.

Comme les précédentes, la commande `wc` travaille sur un fichier.

Sans paramètre, les résultats renvoyés par `wc` sont un peu obscurs. Voyez plutôt :

Code : Console

```
$ wc noms.txt

 8  8 64 noms.txt
```

Ces trois nombres signifient, dans l'ordre :

1. le nombre de lignes.
2. le nombre de mots.
3. le nombre d'octets.

Il fallait le savoir !



Dans le cas de notre fichier `noms.txt`, il est normal d'avoir autant de lignes que de mots car nous avions mis un seul mot par ligne.

`-l` : compter le nombre de lignes

Pour avoir uniquement le nombre de lignes, utilisez `-l` :

Code : Console

```
$ wc -l noms.txt

8 noms.txt
```

`-w` : compter le nombre de mots

Combien de mots différents y a-t-il dans le fichier ?

Code : Console

```
$ wc -w noms.txt

8 noms.txt
```

`-c` : compter le nombre d'octets

Combien d'octets comporte le fichier ?

Code : Console

```
$ wc -c noms.txt

64 noms.txt
```

`-m` : compter le nombre de caractères

Ah, voilà une information qui ne nous a pas été donnée lorsque nous avons lancé la commande `wc` sans paramètre.

L'option `-m` renvoie le nombre de caractères :

Code : Console

```
$ wc -m noms.txt

62 noms.txt
```

Comme vous pouvez le voir, le nombre de caractères est différent du nombre d'octets.

uniq : supprimer les doublons

Parfois, certains fichiers contiennent des lignes en double et on aimerait pouvoir les détecter ou les supprimer. La commande `uniq` est toute indiquée pour cela.

Nous devons travailler sur un fichier **trié**. En effet, la commande `uniq` ne repère que les lignes successives qui sont identiques. Je vous propose de créer un fichier `doublons.txt` contenant les noms suivants :

Code : Console

```
Albert
François
François
François
Jean
jonathan
Marcel
Marcel
patrice
Stéphane
Vincent
```

Il y a des noms en double (et même en triple) dans ce fichier. Appliquons un petit coup de `uniq` là-dessus pour voir ce qu'il en reste :

Code : Console

```
$ uniq doublons.txt

Albert
François
Jean
jonathan
Marcel
patrice
Stéphane
Vincent
```

La liste de noms sans les doublons s'affiche alors dans la console !

Vous pouvez demander à ce que le résultat sans doublons soit écrit dans un autre fichier plutôt qu'affiché dans la console :

Code : Console

```
uniq doublons.txt sans_doublons.txt
```

La liste sans doublons sera écrite dans `sans_doublons.txt`.

-c : compter le nombre d'occurrences

Avec `-c`, la commande `uniq` vous affiche le nombre de fois que la ligne est présente dans le fichier :

Code : Console

```
$ uniq -c doublons.txt
      1 Albert
      3 François
      1 Jean
      1 jonathan
      2 Marcel
      1 patrice
      1 Stéphane
      1 Vincent
```

On sait ainsi qu'il y a trois fois « François », une fois « Jean », deux fois « Marcel », etc.

-d : afficher uniquement les lignes présentes en double

L'option `-d` demande à afficher uniquement les lignes présentes en double :

Code : Console

```
$ uniq -d doublons.txt

François
Marcel
```

Comme seuls François et Marcel avaient des doublons, on les voit ici s'afficher dans la console.



Comme pour les autres commandes présentées dans ce chapitre, je ne vous ai pas fait la liste de toutes les options disponibles. J'ai choisi de vous présenter celles qui me paraissaient les plus intéressantes ou les plus utiles, mais c'est tout à fait subjectif. Ayez le réflexe d'aller regarder le manuel (`man uniq` par exemple) pour connaître la liste exhaustive des options de la commande.

cut : couper une partie du fichier

Vous avez déjà coupé du texte dans un éditeur de texte, non ?
La commande `cut` vous propose de faire cela au sein d'un fichier afin de conserver uniquement une partie de chaque ligne.

Couper selon le nombre de caractères

Par exemple, si vous souhaitez conserver uniquement les caractères 2 à 5 de chaque ligne du fichier, vous taperez :

Code : Console

```
$ cut -c 2-5 noms.txt

ran
arce
lber
ean
tép
atri
ince
onat
```



`cut` a quelques soucis avec les mots contenant des accents. Comme vous pouvez le voir, certains mots ici coupés ont quatre lettres (comme prévu) alors que d'autres en ont trois.
Ceci est dû à l'encodage des caractères, aux accents. La commande `cut` se base sur le nombre d'octets, et comme nous l'avons vu plus tôt, celui-ci n'est pas forcément égal au nombre de caractères. À l'heure actuelle on ne peut rien y faire, c'est la commande `cut` qui devra être mise à jour par les programmeurs.

Pour conserver du 1er au 3ème caractère :

Code : Console

```
$ cut -c -3 noms.txt

Fra
Mar
Alb
```

```
Jea
St
pat
Vin
jon
```

Comme vous pouvez le voir, si on ne met pas de chiffre au début, `cut` comprend que vous voulez parler du premier caractère.

De même, pour conserver du 3ème au dernier caractère :

Code : Console

```
$ cut -c 3- noms.txt

ançois
rcel
bert
an
éphane
trice
ncent
nathan
```

Là encore, pas besoin de donner le numéro du dernier caractère, la commande `cut` comprend comme une grande qu'elle doit couper jusqu'à la fin.

Couper selon un délimiteur

Faisons maintenant quelque chose de bien plus intéressant. Plutôt que de s'amuser à compter le nombre de caractères, nous allons travailler avec ce que l'on appelle un **délimiteur**.

Prenons un cas pratique : les fichiers CSV (*Comma Separated Values.* Ce sont des fichiers dont les valeurs sont séparées par des virgules. Notez qu'Excel utilise plutôt le point-virgule comme séparateur, mais le principe reste le même.) Vous en avez peut-être déjà vu : ils sont générés par des tableurs — tels qu'Excel ou Calc — pour faciliter l'échange et le traitement de données.

Imaginons que vous ayez une (petite) classe et que vous rendiez les notes du dernier contrôle. Vous avez fait un joli tableur et vous avez enregistré le document au format CSV. Le fichier sur lequel nous allons nous baser sera le suivant :

Code : Console

```
Fabrice,18 / 20,Excellent travail
Mathieu,3 / 20,Nul comme d'hab
Sophie,14 / 20,En nette progression
Mélanie,9 / 20,Allez presque la moyenne !
Corentin,11 / 20,Pas mal mais peut mieux faire
Albert,20 / 20,Toujours parfait
Benoît,5 / 20,En grave chute
```

Comme le nom CSV l'indique, les virgules servent à séparer les colonnes. Ces dernières contiennent, dans l'ordre :

- le prénom ;
- la note ;
- un commentaire.

C'est un exemple tout à fait fictif, bien entendu. ;-)

Créez, avec le texte que je viens de vous donner, un nouveau fichier que vous appellerez par exemple `notes.csv`.

Imaginons que nous souhaitions extraire de ce fichier la liste des prénoms. Comment nous y prendrions-nous ?
On ne peut pas utiliser la technique qu'on vient d'apprendre car les prénoms ne font pas tous la même longueur. Nous allons donc nous servir du fait que nous savons que la virgule sépare les différents champs dans ce fichier.

Vous allez avoir besoin d'utiliser deux paramètres :

- `-d` : indique quel est le délimiteur dans le fichier ;
- `-f` : indique le numéro du ou des champs à couper.

Dans notre cas, le délimiteur qui sépare les champs est la virgule.
Le numéro du champ à couper est 1 (c'est le premier).

Testez donc ceci :

Code : Console

```
$ cut -d , -f 1 notes.csv

Fabrice
Vincent
Sophie
Mélanie
Corentin
Albert
Benoît
```

C'est pas beau, ça ? :-)

Après le `-d`, nous avons indiqué quel était le délimiteur (à savoir la virgule « `,` »).
Après le `-f`, nous avons indiqué le numéro du champ à conserver (le premier).

Si nous voulons juste les commentaires :

Code : Console

```
$ cut -d , -f 3 notes.csv

Excellent travail
Nul comme d'hab
En nette progression
Allez presque la moyenne !
Pas mal mais peut mieux faire
Toujours parfait
En grave chute
```

Pour avoir les champs n°1 et n°3 (le prénom et le commentaire) :

Code : Console

```
$ cut -d , -f 1,3 notes.csv

Fabrice,Excellent travail
Vincent,Nul comme d'hab
Sophie,En nette progression
Mélanie,Allez presque la moyenne !
Corentin,Pas mal mais peut mieux faire
Albert,Toujours parfait
Benoît,En grave chute
```

 De même, il est possible de conserver toute une série de champs avec le tiret comme tout à l'heure : `cut -d , -f 2-4 notes.csv` a pour effet de conserver les champs n°
2, 3 et 4.
D'autre part, `cut -d , -f 3- notes.csv` conserve les champs du n°3 jusqu'à la fin.

Vous êtes bien obligés d'admettre que, quand on sait bien s'en servir, la console de Linux peut vous permettre d'effectuer des opérations vraiment puissantes que vous ne pensiez même pas pouvoir faire aussi simplement jusqu'à présent. ;-)

En résumé

- `grep` est une commande couramment utilisée pour rechercher un mot dans un fichier.
- On peut utiliser des expressions régulières, un système complexe mais puissant, pour effectuer des recherches précises. On fait dans ce cas appel à la commande `egrep`.
- `sort` trie des lignes de texte par ordre alphabétique. Le paramètre `-n` permet de trier par ordre numérique.
- `wc` compte le nombre de lignes, de mots et d'octets dans un fichier.
- `uniq` supprime les doublons d'un fichier.
- `cut` coupe une partie d'un fichier.

 Les flux de redirection

Vous devriez maintenant avoir l'habitude d'un certain nombre de commandes que propose la console de Linux. Le fonctionnement est toujours le même :

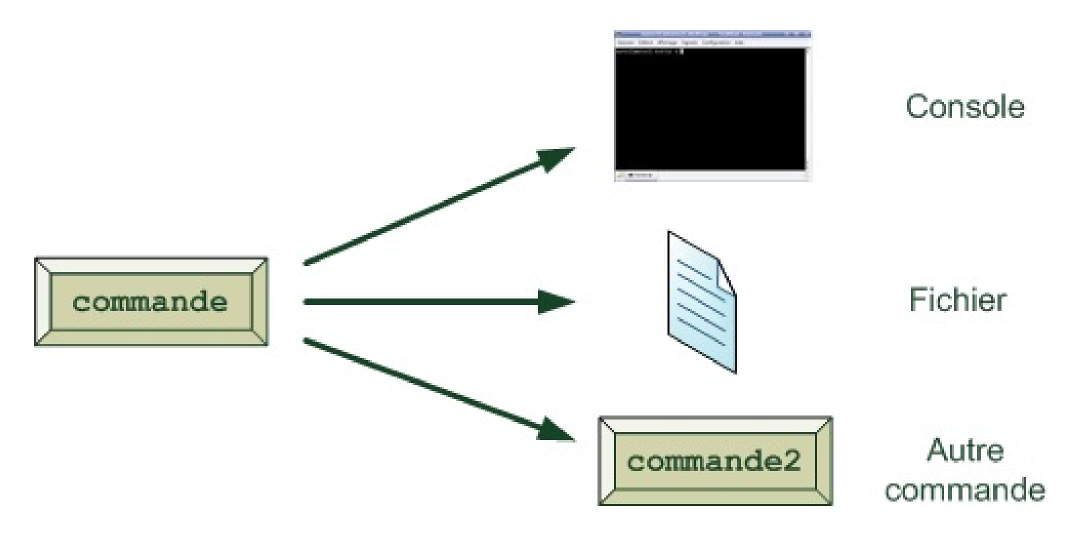
1. vous tapez la commande (par exemple `ls`) ;
2. le résultat s'affiche dans la console.

Ce que vous ne savez pas encore, c'est qu'**il est possible de rediriger ce résultat**. Au lieu que celui-ci s'affiche dans la console, vous allez pouvoir l'envoyer ailleurs : dans un fichier ou en entrée d'une autre commande pour effectuer des « chaînes de commandes ».

Grâce à ce chapitre sur les flux de redirection, vous allez beaucoup gagner en maîtrise de la ligne de commandes !

Dans ce chapitre, nous allons découvrir qu'il est possible de rediriger le résultat d'une commande ailleurs que dans la console.
Où ? Dans un fichier, ou en entrée d'une autre commande pour « chaîner des commandes ». Ainsi, le résultat d'une commande peut en déclencher une autre !
Comment ? À l'aide de petits symboles spéciaux, appelés *flux de redirection*, que vous allez découvrir dans ce chapitre.

Le principe peut être résumé dans le schéma de la figure suivante.



Jusqu'ici, nous n'avons donc exploité que la première possibilité (celle par défaut) : afficher le résultat dans la console. Il nous reste donc bien d'autres techniques à découvrir !



Les flux de redirection constituent une composante essentielle de la console sous Linux et ce, depuis l'époque d'Unix. Ils vont très certainement changer votre façon de « voir » comment la console fonctionne et démultiplier votre contrôle sur les commandes que vous lancez. C'est dire si ce chapitre est important !

Je vais donc d'abord vous demander d'être encore plus attentifs que d'habitude. Non pas que le chapitre soit réellement « compliqué », mais il **doit** bien être compris pour que vous puissiez suivre le reste du livre convenablement.
Au pire des cas, vous pourrez toujours revenir lire ce chapitre si vous avez un trou de mémoire sur les notions que vous y avez apprises. ;-)

> et >> : rediriger le résultat dans un fichier

La manipulation la plus simple que nous allons voir va nous permettre d'**écrire le résultat d'une commande dans un fichier**, au lieu de l'afficher bêtement dans la console.

Préparatifs

Prenons une commande au hasard. Vous vous souvenez de `cut`, que nous avons appris dans le chapitre précédent ?

Nous avions travaillé sur un petit fichier de type « CSV » que les tableurs peuvent générer.
Ce sont les notes des élèves d'une classe à un contrôle :

Code : Console

```
Fabrice,18 / 20,Excellent travail
Mathieu,3 / 20,Nul comme d'hab'
Sophie,14 / 20,En nette progression
Mélanie,9 / 20,Allez presque la moyenne !
Corentin,11 / 20,Pas mal mais peut mieux faire
Albert,20 / 20,Toujours parfait
Benoît,5 / 20,En grave chute
```



Si vous ne l'aviez pas déjà fait dans le chapitre précédent, je vous recommande d'enregistrer ce fichier dans un éditeur de texte (comme Nano) en récupérant le contenu ci-dessus à l'aide du code web. Enregistrez le tout sous le nom `notes.csv`.

La commande `cut` nous avait permis de « couper » une partie du fichier et d'afficher le résultat dans la console. Par exemple, nous avions demandé à `cut` de prendre tout ce qui se trouvait avant la première virgule afin d'avoir la liste des noms de tous les élèves présents à ce contrôle :

Code : Console

```
$ cut -d , -f 1 notes.csv

Fabrice
Vincent
Sophie
Mélanie
Corentin
Albert
Benoît
```

Ce résultat s'est affiché dans la console. C'est ce que font toutes les commandes par défaut… à moins que l'on utilise un flux de redirection !

> : rediriger dans un nouveau fichier

Supposons que nous souhaitions écrire la liste des prénoms dans un fichier, afin de garder sous le coude la liste des élèves présents au contrôle.
C'est là qu'intervient le petit symbole magique > (appelé **chevron**) que je vous laisse trouver sur votre clavier (ceux qui font du HTML le connaissent bien. ;-).

Ce symbole permet de rediriger le résultat de la commande dans le fichier de votre choix. Essayez par exemple de taper ceci :

Code : Console

```
cut -d , -f 1 notes.csv > eleves.txt
```

Regardez la fin de la commande. J'y ai rajouté la petite flèche > qui redirige la sortie de la commande dans un fichier.
Normalement, si vous exécutez cette commande, **rien ne s'affichera dans la console**. Tout aura été redirigé dans un fichier appelé `eleves.txt` qui vient d'être créé pour l'occasion dans le dossier dans lequel vous vous trouviez.



Je le rappelle au cas où : sous Linux, on se moque pas mal de l'extension des fichiers. J'aurais très bien pu créer un fichier sans extension appelé `eleves`. Ici j'ai rajouté un « `.txt` » pour ne pas dérouter ceux qui viennent de Windows, mais il faudra vous habituer à travailler avec des noms de fichiers parfois sans extension.

Faites un petit `ls` (ou `ls -l`, comme vous préférez) pour voir si le fichier est bien présent dans le dossier :

Code : Console

```
$ ls -l
total 20
-rw-r--r-- 1 mateo21 mateo21  91 2008-04-19 19:36 doublons.txt
-rw-r--r-- 1 mateo21 mateo21  56 2008-09-26 12:01 eleves.txt
-rw-r--r-- 1 mateo21 mateo21  35 2008-04-19 17:06 fichier_trie.txt
-rw-r--r-- 1 mateo21 mateo21  20 2008-04-19 19:03 nombres.txt
-rw-r--r-- 1 mateo21 mateo21 253 2008-09-26 12:01 notes.csv
```

Comme vous pouvez le voir, un fichier vient bien d'être créé !
Vous pouvez l'ouvrir avec Nano ou encore l'afficher dans la console avec la commande `cat` (pour afficher tout d'un coup s'il est court) ou `less` (pour afficher page par page s'il est long).



Attention : si le fichier existait déjà il sera écrasé sans demande de confirmation !



Parfois, vous ne voulez ni voir le résultat d'une commande ni le stocker dans un fichier. Dans ce cas, l'astuce consiste à rediriger le résultat dans `/dev/null`. C'est un peu le « trou noir » de Linux : tout ce qui va là-dedans disparaît immédiatement.
Par exemple : `commande_bavarde > /dev/null`

>> : rediriger à la fin d'un fichier

Le double chevron >> sert lui aussi à rediriger le résultat dans un fichier, mais cette fois à la fin de ce fichier.

Avantage : vous ne risquez pas d'écraser le fichier s'il existe déjà. Si le fichier n'existe pas, il sera créé automatiquement.

Normalement, vous devriez avoir créé un fichier `eleves.txt` lors des manipulations précédentes. Si vous faites :

Code : Console

```
cut -d , -f 1 notes.csv >> eleves.txt
```

… les noms seront ajoutés à la fin du fichier, sans écraser le résultat précédent.

Bon, du coup, on a des noms en double maintenant :

Code : Console

```
$ cat eleves.txt
Fabrice
Mathieu
Sophie
Mélanie
Corentin
Albert
Benoît
Fabrice
Mathieu
Sophie
Mélanie
Corentin
Albert
```

```
Benoît
```

Heureusement, vous connaissez les commandes `sort` et `uniq` qui peuvent vous permettre de faire un peu de ménage là-dedans. Je vous laisse supprimer les doublons.
N'oubliez pas qu'il faut que le fichier soit trié pour que la commande `uniq` fonctionne !



Quand utilise-t-on le double chevron pour mettre le résultat à la fin d'un fichier ?

Personnellement, j'ai des commandes qui s'exécutent automatiquement à certaines heures (on verra comment faire ça plus tard). Comme je ne suis pas devant mon ordinateur lorsque ces commandes s'exécutent, j'enregistre un log de ce qui s'est passé dans un fichier :

Code : Console

```
macommande >> resultats.log
```

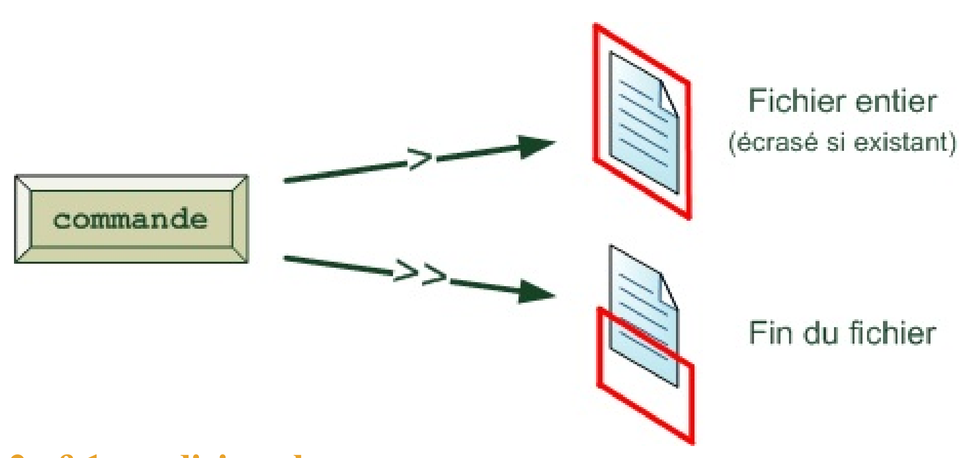
Grâce à ça, si j'ai un doute sur ce qui a pu se passer lors de l'exécution d'une commande, je n'ai qu'à consulter le fichier `resultats.log`.

Résumé

Nous venons de découvrir deux flux de redirection dans des fichiers :

- > : redirige dans un fichier et l'écrase s'il existe déjà ;
- >> : redirige à la fin d'un fichier et le crée s'il n'existe pas.

Le schéma de la figure suivante récapitule ce que nous venons de voir.



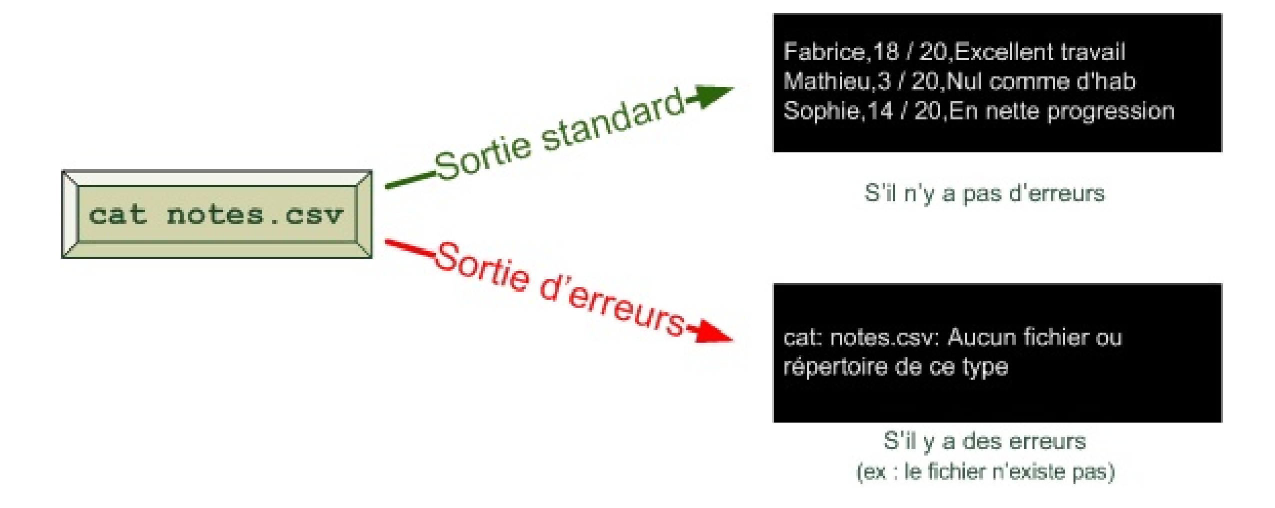
2>, 2>> et 2>&1 : rediriger les erreurs

Allons un peu plus loin. Il faut savoir que toutes les commandes produisent deux flux de données différents, comme le montre la figure suivante :

- **la sortie standard** : pour tous les messages (sauf les erreurs) ;
- **la sortie d'erreurs** : pour toutes les erreurs.

Prenons un exemple concret pour voir comment ça se passe.
Supposons que vous fassiez un `cat` du fichier `notes.csv` pour afficher son contenu. Il y a deux possibilités :

- **si tout va bien**, le résultat (le contenu du fichier) s'affiche sur la sortie standard ;
- **s'il y a une erreur**, celle-ci s'affiche dans la sortie d'erreurs.



Par défaut, tout s'affiche dans la console : la sortie standard comme la sortie d'erreurs. Cela explique pourquoi vous ne faisiez pas la différence entre ces deux sorties jusqu'ici : elles avaient l'air identiques.

Tout à l'heure, nous avons vu comment rediriger la sortie standard dans un fichier. Toutefois, les erreurs continuent d'être affichées dans la console. Faites le test :

Code : Console

```
cut -d , -f 1 fichier_inexistant.csv > eleves.txt
cut: fichier_inexistant.csv: Aucun fichier ou répertoire de ce type
```

Le fichier `fichier_inexistant.csv` n'existe pas (comme son nom l'indique). L'erreur s'est affichée dans la console au lieu d'avoir été envoyée dans `eleves.txt`.

Rediriger les erreurs dans un fichier à part

On pourrait souhaiter enregistrer les erreurs dans un fichier à part pour ne pas les oublier et pour pouvoir les analyser ensuite.

Pour cela, on utilise l'opérateur `2>`. Vous avez bien lu : c'est le chiffre 2 collé au chevron que nous avons utilisé tout à l'heure.

Faisons une seconde redirection à la fin de cette commande `cut` :

Code : Console

```
cut -d , -f 1 fichier_inexistant.csv > eleves.txt 2> erreurs.log
```

Il y a deux redirections ici :

- `> eleves.txt` : redirige le résultat de la commande (sauf les erreurs) dans le fichier `eleves.txt`. C'est la sortie standard ;
- `2> erreurs.log` : redirige les erreurs éventuelles dans le fichier `erreurs.log`. C'est la sortie d'erreurs.

Vous pouvez vérifier : si `fichier_inexistant.csv` n'a pas été trouvé, l'erreur aura été inscrite dans le fichier `erreurs.log` au lieu d'être affichée dans la console.


Notez qu'il est aussi possible d'utiliser `2>>`

pour ajouter les erreurs à la fin du fichier.

Fusionner les sorties

Parfois, on n'a pas envie de séparer les informations dans deux fichiers différents. Heureusement, il est possible de fusionner les sorties dans un seul et même fichier. Comment ?

Il faut utiliser le code suivant : `2>&1`.
Cela a pour effet de rediriger toute la sortie d'erreurs dans la sortie standard. Traduction pour l'ordinateur : « envoie les erreurs au même endroit que le reste ».

Essayez donc ceci :

Code : Console

```
cut -d , -f 1 fichier_inexistant.csv > eleves.txt 2>&1
```

Tout ira désormais dans `eleves.txt` : le résultat (si cela a fonctionné), de même que les erreurs (s'il y a eu un problème).

Petite subtilité : je vous ai dit tout à l'heure qu'il était possible de faire `2>>` pour rediriger les erreurs à la fin d'un fichier d'erreurs. Toutefois, il n'est pas possible d'écrire `2>>&1`. Essayez, ça ne marchera pas.

En fait, le symbole `2>&1` va envoyer les erreurs dans le même fichier et **de la même façon** que la sortie standard. Donc, si vous écrivez :

Code : Console

```
cut -d , -f 1 fichier_inexistant.csv >> eleves.txt 2>&1
```

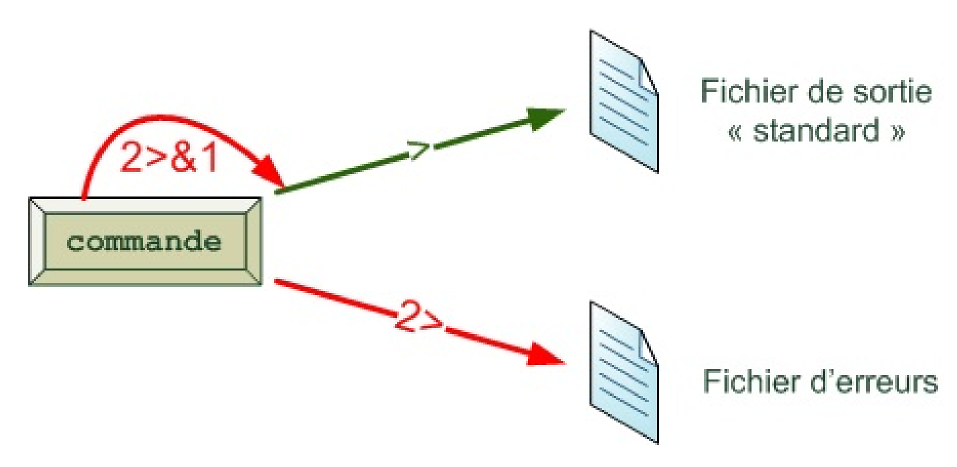
… les erreurs seront **ajoutées** à la fin du fichier `eleves.txt` comme le reste des messages.

Résumé

Nous avons découvert trois symboles :

- `2>` : redirige les erreurs dans un fichier (s'il existe déjà, il sera écrasé) ;
- `2>>` : redirige les erreurs à la fin d'un fichier (s'il n'existe pas, il sera créé) ;
- `2>&1` : redirige les erreurs au même endroit et de la même façon que la sortie standard.

Le tout est illustré sur la figure suivante.



Comprenez-vous bien ce schéma ?

On peut choisir de rediriger les erreurs dans un fichier à part (avec `2>`) ou bien de les rediriger au même endroit que la sortie standard (avec `2>&1`).



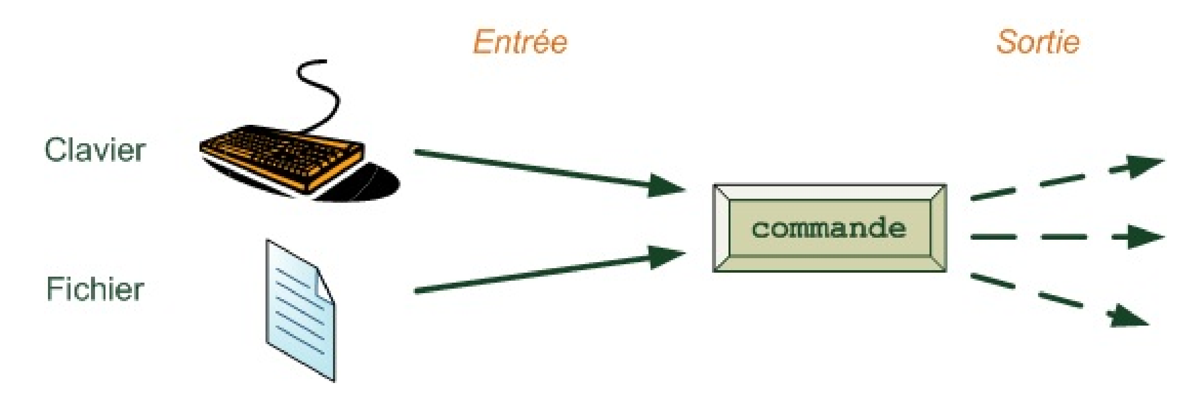
J'ai volontairement omis de parler sur ce schéma de >>

et de 2>> afin de ne pas le surcharger, mais le principe est le même sauf qu'on ajoute à la fin d'un fichier au lieu de l'écraser.

< et << : lire depuis un fichier ou le clavier

Pour le moment, nous avons redirigé uniquement la **sortie** des commandes. Nous avons décidé où envoyer les messages issus de ces commandes.

Maintenant, je vous propose de faire un peu l'inverse, c'est-à-dire de décider d'où vient l'**entrée** d'une commande. Jusqu'alors, l'entrée venait des paramètres de la commande… mais on peut faire en sorte qu'elle vienne d'un fichier ou d'une saisie au clavier ! Regardez l'illustration de la figure suivante.



< : lire depuis un fichier

Le chevron ouvrant < (à ne pas confondre avec le chevron fermant que nous avons utilisé tout à l'heure) permet d'indiquer d'où vient l'entrée qu'on envoie à la commande.

On va prendre un exemple tout bête : la commande `cat`.

Code : Console

```
cat < notes.csv
```

Cela aura pour effet d'afficher le contenu du fichier envoyé en entrée :

Code : Console

```
$ cat < notes.csv
Fabrice,18 / 20,Excellent travail
Mathieu,3 / 20,Nul comme d'hab'
Sophie,14 / 20,En nette progression
Mélanie,9 / 20,Allez presque la moyenne !
Corentin,11 / 20,Pas mal mais peut mieux faire
Albert,20 / 20,Toujours parfait
Benoît,5 / 20,En grave chute
```



Il n'y a rien d'extraordinaire.
On ne faisait pas pareil avant en écrivant juste `cat notes.csv` par hasard ?

Si. Écrire `cat < notes.csv` est strictement identique au fait d'écrire `cat notes.csv`… du moins en apparence. Le résultat produit est le même, mais ce qui se passe derrière est très différent.

- Si vous écrivez `cat notes.csv`, la commande `cat` reçoit en entrée le nom du fichier `notes.csv` qu'elle doit ensuite se charger d'ouvrir pour afficher son contenu.
- Si vous écrivez `cat < notes.csv`, la commande `cat` reçoit **le contenu** de `notes.csv` qu'elle se contente simplement d'afficher dans la console. C'est le shell (le programme qui gère la console) qui se charge d'envoyer le contenu de `notes.csv` à la commande `cat`.

Bref, ce sont deux façons de faire la même chose mais de manière très différente.

Pour le moment, je n'ai pas d'exemple plus intéressant à vous proposer à ce sujet, mais retenez cette possibilité car vous finirez par en avoir besoin, faites-moi confiance. ;-)

<< : lire depuis le clavier progressivement

Le double chevron ouvrant << fait quelque chose d'assez différent : il vous permet d'envoyer un contenu à une commande avec votre clavier.
Cela peut s'avérer très utile. Je vous propose un exemple concret pour bien voir ce que ça permet de faire en pratique.

Essayez de taper ceci :

Code : Console

```
sort -n << FIN
```

La console vous propose alors de taper du texte.

Code : Console

```
$ sort -n << FIN
>
```

Comme `sort -n` sert à trier des nombres, on va justement écrire des nombres, un par ligne (en appuyant sur la touche `Entrée` à chaque fois).

Code : Console

```
$ sort -n << FIN
> 13
> 132
> 10
> 131
```

Continuez ainsi jusqu'à ce que vous ayez terminé.

Lorsque vous avez fini, tapez `FIN` pour arrêter la saisie.
Tout le texte que vous avez écrit est alors envoyé à la commande (ici `sort`) qui traite cela en entrée. Et, comme vous pouvez vous en douter, la commande `sort` nous trie nos nombres !

Code : Console

```
$ sort -n << FIN
> 13
> 132
```

```
> 10
> 131
> 34
> 87
> 66
> 68
> 65
> FIN
10
13
34
65
66
68
87
131
132
```

Sympa, non ?
Cela vous évite d'avoir à créer un fichier si vous n'en avez pas besoin.

Vous pouvez faire la même chose avec une autre commande, comme par exemple `wc` pour compter le nombre de mots ou de caractères.

Code : Console

```
$ wc -m << FIN
> Combien de caractères dans cette phrase ?
> FIN
42
```



Une question : ce mot `FIN` est-il obligatoire ?

Non, vous pouvez le remplacer par ce que vous voulez.
Lorsque vous tapez la commande, vous pouvez utiliser le mot que vous voulez. Par exemple :

Code : Console

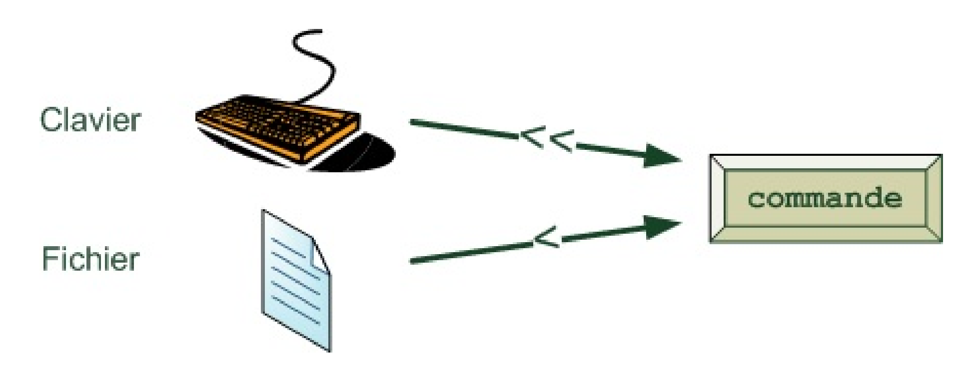
```
$ wc -m << STOP
> Combien de caractères dans cette phrase ?
> STOP
42
```

Ce qui compte, c'est que vous définissiez un mot-clé qui servira à indiquer la fin de la saisie.
Notez par ailleurs que rien ne vous oblige à écrire ce mot en majuscules.

Résumé

Nous pouvons donc « alimenter » des commandes de deux manières différentes, comme le montre la figure suivante :

- < : envoie le contenu d'un fichier à une commande ;
- << : passe la console en mode saisie au clavier, ligne par ligne. Toutes ces lignes seront envoyées à la commande lorsque le mot-clé de fin aura été écrit.



Vous pouvez tout à fait combiner ces symboles avec ceux qu'on a vus précédemment. Par exemple :

Code : Console

```
$ sort -n << FIN > nombres_tries.txt 2>&1
> 18
> 27
> 1
> FIN
```

Les nombres saisis au clavier seront envoyés à `nombres_tries.txt`, de même que les erreurs éventuelles.

Hé, mine de rien, on commence à rédiger là des commandes assez complexes !
Mais vous allez voir, on peut faire encore mieux.

| : chaîner les commandes

Passons maintenant au symbole le plus intéressant que vous utiliserez le plus souvent : le **pipe** `|` (prononcez « païpe », comme un bon Anglais). Son but ? Chaîner des commandes.

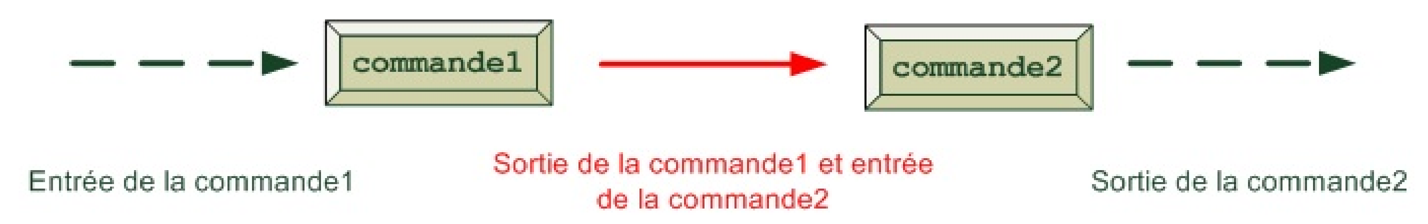


Le pipe `|` n'est pas un symbole qu'on a l'habitude d'écrire. Pourtant, il y en a forcément un sur votre clavier (parfois représenté sous la forme d'une ligne verticale en pointillés).
Sur un clavier AZERTY français par exemple, vous pouvez l'écrire en combinant les touches `Alt Gr + 6` et sur un clavier belge, `Alt Gr + 1`. Sur un clavier Mac, c'est `Alt + Shift + L`

.

La théorie

« Chaîner des commandes » ? Cela signifie connecter la sortie d'une commande à l'entrée d'une autre commande (comme le montre la figure suivante).



En gros, **tout ce qui sort de la `commande1` est immédiatement envoyé à la `commande2`**. Et vous pouvez chaîner des commandes comme cela indéfiniment !

Cette fonctionnalité est vraiment une des plus importantes et décuple littéralement les possibilités offertes par la console.
Souvenez-vous : dans le chapitre précédent, **je vous disais que chaque commande Unix avait un et un seul rôle, mais qu'elle le remplissait bien**. Parfois, l'utilité de certaines commandes seules peut paraître limitée, mais celles-ci prennent en général tout leur sens lorsqu'on les combine à d'autres commandes.

La pratique

Voyons quelques cas concrets (on pourrait trouver une infinité d'exemples).

Trier les élèves par nom

Si vous vous souvenez bien, nous avons toujours un fichier `notes.csv` qui contient la liste des élèves et leurs notes :

Code : Console

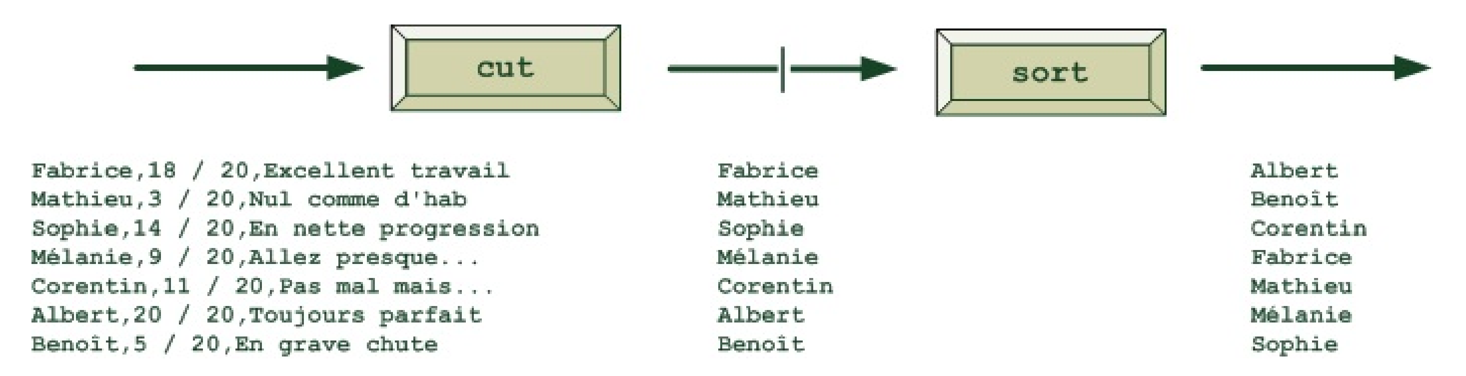
```
Fabrice,18 / 20,Excellent travail
Mathieu,3 / 20,Nul comme d'hab'
Sophie,14 / 20,En nette progression
Mélanie,9 / 20,Allez presque la moyenne !
Corentin,11 / 20,Pas mal mais peut mieux faire
Albert,20 / 20,Toujours parfait
Benoît,5 / 20,En grave chute
```

Avec `cut`, on peut récupérer les noms. Avec `sort`, on peut les trier par ordre alphabétique. Pourquoi ne pas connecter `cut` à `sort` pour avoir la liste des noms triés ?

Code : Console

```
$ cut -d , -f 1 notes.csv | sort
Albert
Benoît
Corentin
Fabrice
Mathieu
Mélanie
Sophie
```

Le pipe effectue la connexion entre la sortie de `cut` (des noms dans le désordre) et l'entrée de `sort`, comme l'illustre la figure suivante.





On peut même aller plus loin et écrire cette liste triée dans un fichier :

Code : Console

```
cut -d , -f 1 notes.csv | sort > noms_tries.txt
```

Trier les répertoires par taille

La commande du permet d'obtenir la taille de chacun des sous-répertoires du répertoire courant (je vous conseille de vous placer dans votre home en tapant d'abord cd) :

Code : Console

```
$ du
4        ./.gnome2_private
40        ./.local/share/Trash/files
4        ./.local/share/Trash/info
12       ./.local/share/Trash
160       ./.local/share
20       ./.local
...
```

Deux problèmes : cette liste est parfois très longue et n'est pas triée.

Un problème à la fois. Tout d'abord, on aimerait par exemple avoir cette même liste dans l'ordre décroissant de taille des répertoires pour repérer plus facilement les plus gros d'entre eux qui prennent de la place sur notre disque.

Pour avoir cette liste du plus grand au plus petit, il nous suffit d'écrire :

Code : Console

```
du | sort -nr
```

On envoie tout le contenu de du à sort qui se charge de trier les nombres au début de chacune des lignes.

Code : Console

```
$ du | sort -nr

...

4        ./.evolution/memos/config
4        ./.evolution/calendar/config
4        ./.evolution/cache
4        ./bin
```

Problème : comme les plus gros répertoires ont été affichés en premier, et que j'ai beaucoup de sous-répertoires, je dois remonter très haut dans la console pour retrouver les plus gros d'entre eux.

Que diriez-vous de connecter cette sortie à head ? Cette commande permet de filtrer uniquement les premières lignes qu'elle reçoit, nous l'avons déjà étudiée dans un chapitre précédent.

Code : Console

```
$ du | sort -nr | head
120920   .
59868    ./.ies4linux
43108    ./.ies4linux/ie6
41360    ./.ies4linux/ie6/drive_c
41248    ./.ies4linux/ie6/drive_c/windows
40140    ./Desktop
34592    ./.ies4linux/ie6/drive_c/windows/system32
16728    ./.ies4linux/downloads
13128    ./.mozilla
13124    ./.mozilla/firefox
```

Vous pouvez paramétrer le nombre de résultats affichés avec l'option `-n` de `head`. Si vous avez oublié comment l'utiliser, retournez lire le cours sur `head`, ou consultez le manuel.

Si vous voulez naviguer à travers tous les résultats, vous pouvez connecter la sortie à `less`. Cette commande permet d'afficher des résultats page par page ; ça nous est justement utile dans le cas présent où nous avons beaucoup de résultats !

Code : Console

```
du | sort -nr | less
```

Essayez !
Vous allez vous retrouver avec un affichage de `less`, page par page.

Code : Console

```
120920  .
59868   ./.ies4linux
43108   ./.ies4linux/ie6
41360   ./.ies4linux/ie6/drive_c
41248   ./.ies4linux/ie6/drive_c/windows
40140   ./Desktop
34592   ./.ies4linux/ie6/drive_c/windows/system32
16728   ./.ies4linux/downloads
13128   ./.mozilla
13124   ./.mozilla/firefox
13112   ./.mozilla/firefox/v5p4a55d.default
12604   ./.ies4linux/downloads/ie6
11808   ./.ies4linux/downloads/ie6/FR
5848    ./.mozilla/firefox/v5p4a55d.default/Cache
3656    ./.ies4linux/ie6/drive_c/windows/profiles
3616    ./.ies4linux/ie6/drive_c/windows/profiles/mateo21
3496    ./.ies4linux/ie6/drive_c/windows/profiles/mateo21/Local Settings
3416    ./.ies4linux/ie6/drive_c/windows/profiles/mateo21/Local Settings/Temporary
3408    ./.ies4linux/ie6/drive_c/windows/profiles/mateo21/Local Settings/Temporary
2220    ./.ies4linux/ie6/drive_c/windows/fonts
2012    ./ies4linux-2.99.0.1
:
```

Vous pouvez maintenant voir les premiers fichiers (les plus gros) et descendre progressivement vers les fichiers plus petits, page par page avec la touche `Espace` ou ligne par ligne, avec la touche `Entrée` (ou les flèches du clavier).

Exercice : peut-être avez-vous toujours trop de répertoires sous les yeux et que vous vous intéressez seulement à certains d'entre eux. Pourquoi ne pas filtrer les résultats avec `grep`, pour afficher uniquement la taille des répertoires liés à... `Firefox` par exemple ?

Lister les fichiers contenant un mot

Allez, un dernier exercice tordu pour finir en beauté.

Avec `grep`, on peut connaître la liste des fichiers contenant un mot dans tout un répertoire (option `-r`). Le problème est que cette sortie est un peu trop **verbeuse** (il y a trop de texte) : il y a non seulement le nom du fichier mais aussi la ligne dans laquelle le mot a été trouvé.

Code : Console

```
/var/log/installer/syslog:Apr  6 15:14:43 ubuntu NetworkManager: <debug> [120749488
```

```
/var/log/installer/syslog:Apr  6 15:23:27 ubuntu python: log-output
```

Heureusement, le nom du fichier et le contenu de la ligne sont séparés par un deux-points. On connaît `cut`, qui permet de récupérer uniquement une partie de la ligne. Il nous permettrait de conserver uniquement le nom du fichier.
Problème : si le même mot a été trouvé plusieurs fois dans un fichier, le fichier apparaîtra en double ! Pour supprimer les doublons, on peut utiliser `uniq`, à condition d'avoir bien trié les lignes avec `sort` auparavant.

Alors, vous avez une petite idée de la ligne qu'il va falloir écrire ?
Je vous propose de rechercher les fichiers qui contiennent le mot « log » dans le dossier `/var/log`. Notez qu'il faudra passer root avec `sudo` pour avoir accès à tout le contenu de ce répertoire.

Voici la commande que je vous propose d'utiliser :

Code : Console

```
sudo grep log -Ir /var/log  | cut -d : -f 1  | sort | uniq
```

Que fait cette commande ?

1. Elle liste tous les fichiers contenant le mot « log » dans `/var/log` (`-I` permettant d'exclure les fichiers binaires).
2. Elle extrait de ce résultat uniquement les noms des fichiers.
3. Elle trie ces noms de fichiers.
4. Elle supprime les doublons.

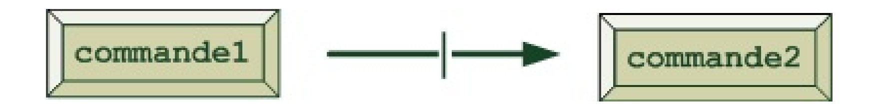
Et voilà le résultat !

Code : Console

```
$ sudo grep log -Ir /var/log  | cut -d : -f 1  | sort | uniq
/var/log/acpid
/var/log/auth.log
/var/log/boot
/var/log/bootstrap.log
/var/log/dist-upgrade/apt-term.log
/var/log/dmesg
/var/log/dmesg.0
/var/log/gdm/
/var/log/installer/partman
/var/log/installer/syslog
/var/log/kern.log.0
/var/log/messages
/var/log/messages.0
/var/log/syslog
/var/log/syslog.0
/var/log/udev
/var/log/Xorg.0.log
/var/log/Xorg.0.log.old
/var/log/Xorg.20.log
/var/log/Xorg.20.log.old
/var/log/Xorg.21.log
```

Résumé

Le résumé est simple, et c'est dans sa simplicité qu'il tire toute sa beauté et sa puissance (non, je ne suis pas fou !), comme l'illustre la figure suivante.



S'il y avait un schéma à retenir, ce serait celui-là. Ça tombe bien, c'est le plus simple.

Je vous laisse vous entraîner avec le pipe, nous le réutiliserons très certainement dans les prochains chapitres. Essayez d'inventer des combinaisons ! ;-)



Les espaces avant et après le pipe ne sont en général pas obligatoires, mais je préfère les mettre ici pour une meilleure lisibilité.

En résumé

- Au lieu d'afficher le résultat d'une commande dans une console, il est possible de l'enregistrer dans un fichier. Il suffit d'ajouter le symbole > suivi du nom du fichier à la fin de la commande. Par exemple `ls > liste_fichiers.txt` enregistre la liste des fichiers dans un fichier plutôt que de l'afficher en console.
- Le symbole>> enregistre à la fin du fichier au lieu de l'écraser s'il existe déjà.
- Les symboles `2>` et `2>>` permettent de rediriger seulement les erreurs dans un fichier. Quant à `2>&1` il redirige les erreurs dans le même fichier que les messages normaux.
- < permet de lire des données depuis un fichier et de les envoyer à une commande, tandis que << lit les données depuis le clavier.
- Le symbole `|` combine des commandes : les données de la commande à sa gauche sont envoyées à la commande à sa droite. Ainsi, `du | sort -nr` récupère la liste des fichiers avec leur taille et l'envoie à `sort` pour qu'il la trie.

 Surveiller l'activité du système

Comme tous les OS actuels, Linux est un système **multi-tâches** : il est capable de gérer plusieurs programmes tournant en même temps.
Mieux encore, Linux est un système **multi-utilisateurs** : plusieurs personnes peuvent utiliser la même machine en même temps (en s'y connectant via Internet).

Tous ces programmes et ces personnes qui sont sur votre PC peuvent vite donner le tournis. Parfois, l'ordinateur peut se retrouver surchargé à cause d'un programme. Qui a lancé ce programme ? Depuis quand ? Comment arrêter un programme qui ne répond plus ?

Sous Windows, vous avez probablement entendu parler de la commande magique `Ctrl + Alt + Suppr` qui peut parfois vous sortir de bien des situations embarrassantes. Sous Linux, on utilise d'autres outils et d'autres techniques que vous allez apprendre à connaître ici.

w : qui fait quoi ?

Nous allons apprendre dans ce chapitre à utiliser une série de commandes qui nous permettront de savoir ce qui se passe actuellement dans notre ordinateur.

La première commande que je veux vous faire découvrir est très courte et facile à retenir : c'est `w` (comme la lettre, oui, oui).

C'est la première commande que je tape en général quand je me connecte à un serveur surchargé et que je veux essayer de comprendre ce qui se passe. Cela me permet de voir d'un seul coup d'oeil si la machine est vraiment surchargée (et si oui, à quel point) et si quelqu'un d'autre est en train d'intervenir sur la machine.



Si vous utilisez Linux sur votre ordinateur personnel, tranquillement chez vous, vous êtes seuls à l'utiliser en ce moment. Pour que d'autres personnes puissent se connecter à votre ordinateur via Internet, il faut avoir configuré Linux pour ça. Nous verrons comment faire cela plus tard. On en a principalement besoin sur les serveurs.

Essayons d'utiliser `w` pour voir comment ça marche ; n'ayez pas peur, c'est sans danger :

Code : Console

```
$ w
 16:50:30 up  8:50,  2 users,  load average: 0,08, 0,34, 0,31
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
mateo21  :0       -                19Apr08 ?xdm?
   3:38m  1.18s /usr/bin/gnome-
mateo21  pts/0    :0.0             16:49    0.00s  0.33s  0.03s w
```

Bon : à première vue, c'est court mais dense, ça n'a pas l'air très clair.

Pourtant, cette commande nous donne un condensé d'informations très utiles que je vais vous présenter dans l'ordre, de gauche à droite et de haut en bas.

L'heure (aussi accessible via `date`)

Ici, l'heure qui nous est donnée est `16:50:30` (16 h 50 mn 30 s).

Cette information est aussi accessible depuis la commande `date` qui nous donne… la date, l'heure et le décalage horaire.

Code : Console

```
$ date
samedi 16 octobre 2010, 17:26:27 (UTC+0200)
```

La commande `date` permet en outre de modifier la date enregistrée dans l'ordinateur. C'est un peu particulier et pas très

intéressant, nous ne verrons donc pas comment le faire ici (mais il vous suffit de lire le manuel si vous en avez vraiment besoin.)

L'*uptime* (aussi accessible via `uptime`)

Dans notre exemple plus haut, l'information d'*uptime* est la suivante : `up 8:50`. C'est la durée de fonctionnement de l'ordinateur.



L'uptime peut aussi être obtenu via la commande `uptime`.

En soi, cette information n'a pas l'air très utile mais elle permet quand même de savoir depuis combien de temps l'ordinateur travaille, et donc depuis combien de temps il n'a pas été redémarré.

Notez que, contrairement à Windows, il est extrêmement rare que l'installation d'un programme nous réclame de redémarrer l'ordinateur. En fait, vous avez besoin de le redémarrer principalement quand vous mettez à jour le noyau (le coeur) de Linux. Sinon, ce n'est jamais nécessaire.

Ce mode de fonctionnement est particulièrement adapté sur les serveurs qui, par définition, sont des machines qui doivent être tout le temps allumées pour servir les gens qui en ont besoin. Par exemple, les serveurs du Site du Zéro qui vous délivrent les pages du site 24 h/24 et 7 j/7 sont tout le temps allumés et nous n'avons pratiquement jamais besoin de les redémarrer. Pour preuve, l'uptime de notre serveur au moment où j'écris ces lignes :

Code : Console

```
$ uptime
 17:45:58 up 211 days, 15:24,  1 user,  load average: 2.44, 2.66, 2.28
```

Notre serveur est en fonctionnement depuis 211 jours. Il n'a pas eu besoin d'être redémarré depuis. Cela témoigne notamment de la robustesse de Linux et de sa capacité à « tenir le coup » pendant très longtemps.

La charge (aussi accessible via `uptime` et `tload`)

En haut à droite de notre exemple, nous avons la charge. Ce sont trois valeurs décimales : `load average: 0,08, 0,34, 0,31`.

La charge est un indice de l'activité de l'ordinateur. Il y a trois valeurs :

1. la première correspond à la charge moyenne depuis 1 minute (0,08) ;
2. la seconde à la charge moyenne depuis 5 minutes (0,34) ;
3. la dernière à la charge moyenne depuis 15 minutes (0,31).



Qu'est-ce que ce nombre représente ?

C'est un peu compliqué. Si vous voulez vraiment savoir, la doc nous dit qu'il s'agit du nombre moyen de processus (programmes) en train de tourner et qui réclament l'utilisation du processeur.

Cela veut dire que, depuis une minute, il y a en moyenne 0,33 processus qui réclament le processeur. Votre processeur est donc actif 33 % du temps.

Mais ce nombre dépend du nombre de processeurs de votre ordinateur. Un ordinateur *dual core* ne sera complètement chargé que lorsque la valeur aura atteint 2. Pour un *quad core* (4 coeurs de processeur), la valeur maximale avant surcharge sera de 4.

Bref, rien ne vous oblige à savoir ce que ce nombre signifie. Vous avez juste besoin de savoir que, lorsqu'il dépasse 1 (si vous avez un processeur), 2 ou 4, alors votre ordinateur est surchargé. J'ai déjà vu des machines avec une charge de 60, et même plus !

Quand la charge est très élevée pendant une longue période, c'est qu'il y a clairement un problème. Il y a trop de programmes qui

réclament le processeur et quelque chose ne va pas dans l'ordinateur. Celui-ci aura du mal à répondre en cas de forte charge.

Notez que vous pouvez obtenir un graphique de l'évolution de la charge en console via la commande `tload`. Le graphe évolue au fur et à mesure du temps, il faut patienter un petit peu avant d'avoir quelque chose, comme l'illustre la figure suivante.

```
 1.38, 1.78, 1.90 -------------------------------------------------------------
                  *       ***    *   *
                  *       *** * *   *  * *         **    *
                  *       ***** *   **** *        ***    *
                  *       ***** *   **** **       **** ***
                  *       ***** *   **** ** *   ****** ***                   *
=--------*------*****-*--****-****   ***********    *                        *
**       *       ***** *   **** *****   ************ **   **       *  *     *****
***      *       ***** *   **** *********************** ***      *****    ***** *
***      *       ***** *   **** ************************** *    ******* ********
***** *  *       ***** *   **** ****************************** ******* ********
******** * ***   ***** **  **** ****************************** ****************
==============================================================================
******************************************************************************
******************************************************************************
******************************************************************************
******************************************************************************
******************************************************************************
========*======*****=*==****=*************************************************
******************************************************************************
******************************************************************************
******************************************************************************
******************************************************************************
******************************************************************************
```

Vous pouvez quitter le graphe avec `Ctrl + C`.

La liste des connectés (aussi accessible via `who`)

Enfin, le tableau en bas qui nous est donné par `w` est surtout intéressant sur un serveur (une machine partagée par plusieurs utilisateurs). Il donne la liste des personnes connectées sur la machine, ce qu'ils sont en train de faire et depuis combien de temps.

Code : Console

```
USER      TTY       FROM                 LOGIN@   IDLE   JCPU   PCPU WHAT
mateo21   :0        -                   19Apr08 ?xdm?
   3:38m  1.18s /usr/bin/gnome
mateo21   pts/0     :0.0                16:49    0.00s  0.33s  0.03s w
```

Là, j'étais sur mon ordinateur personnel sous Ubuntu. Je ne l'ai pas configuré pour qu'on puisse s'y connecter depuis Internet (comme vous, certainement), ce qui explique pourquoi je suis seul.
Certes, j'apparais deux fois. Nous allons comprendre pourquoi lorsque nous aurons appris à lire le tableau.

Il n'est pas nécessaire de décrire chacune des colonnes. Sachez qu'en gros vous avez :

- `USER` : le nom de l'utilisateur (son login) ;
- `TTY` : le nom de la console dans laquelle se trouve l'utilisateur. Souvenez-vous que sous Linux il y a en général six consoles (`tty1` à `tty6`) et qu'en plus de ça, on peut en ouvrir une infinité grâce aux consoles graphiques (leur nom commence par `pts`, en général), comme le propose le programme « Terminal » sous Unity ou « Konsole » sous KDE ;
- `FROM` : c'est l'adresse IP (ou le nom d'hôte) depuis laquelle il se connecte. Ici, comme je me suis connecté en local (sur ma propre machine, sans passer par Internet), il n'y a pas vraiment d'IP ;
- `LOGIN@` : l'heure à laquelle cet utilisateur s'est connecté ;
- `IDLE` : depuis combien de temps cet utilisateur est inactif (depuis combien de temps il n'a pas lancé de commande) ;

- `WHAT` : la commande qu'il est en train d'exécuter en ce moment. En général, si vous voyez `bash`, cela signifie que l'invite de commandes est ouverte et qu'aucune commande particulière n'est exécutée.

Dans mon cas, on voit donc deux utilisateurs (deux fois moi). Le premier correspond à la session « graphique » : on le devine notamment grâce à la dernière colonne `WHAT` qui indique que cet utilisateur est en train d'exécuter l'environnement graphique Gnome.

L'autre utilisateur est sur une console (ici, une console « graphique » lancée depuis Gnome). Cet utilisateur est en train d'exécuter… la commande `w` ! En effet, lorsque je lance `w` je me « vois » en train de l'exécuter dans la liste des utilisateurs connectés, c'est parfaitement normal.

ps & top : lister les processus

La commande `w` nous a permis de faire rapidement le point sur l'état du système. Allons plus loin, maintenant : nous allons apprendre à lister les processus qui tournent sur votre machine.

Pour faire simple, dites-vous qu'un **processus** est un programme qui tourne en mémoire. La plupart des programmes ne font tourner qu'un processus en mémoire (une seule version d'eux-mêmes). C'est le cas d'OpenOffice par exemple. D'autres lancent des copies d'eux-mêmes, c'est le cas du navigateur Google Chrome qui crée autant de processus en mémoire que d'onglets ouverts.



Sur un serveur web, on utilise en général le logiciel Apache qui délivre les pages web aux internautes. Ce logiciel crée beaucoup de processus pour séparer ses activités. Il en va de même pour les systèmes de gestion de bases de données, comme MySQL et PostgreSQL.
Il ne faut pas s'inquiéter si un programme génère beaucoup de processus, cela n'est pas anormal.

Si vous faites la liste des processus qui tournent sur votre machine, vous risquez d'être surpris. Vous en reconnaîtrez certains, mais vous en verrez beaucoup d'autres qui ont été lancés par le système d'exploitation et dont vous n'avez jamais eu connaissance.

Pour lister les processus qui tournent sous Windows, on utilise `Ctrl + Alt + Suppr` et on va dans l'onglet « Processus ». Sous Linux, on peut utiliser deux commandes différentes : `ps` et `top`.

ps : liste statique des processus

`ps` vous permet d'obtenir la liste des processus qui tournent au moment où vous lancez la commande. Cette liste n'est pas actualisée en temps réel, contrairement à ce que fait `top` et qu'on verra plus tard.

Essayons d'utiliser `ps` sans paramètre :

Code : Console

```
$ ps
  PID TTY          TIME CMD
23720 pts/0    00:00:01 bash
29941 pts/0    00:00:00 ps
```

On distingue quatre colonnes.

- `PID` : c'est le numéro d'identification du processus. Chaque processus a un numéro unique qui permet de l'identifier. Ce numéro nous sera utile plus tard lorsque nous voudrons arrêter le processus.
- `TTY` : c'est le nom de la console depuis laquelle a été lancé le processus.
- `TIME` : la durée d'exécution du processus. Plus exactement, cela correspond à la durée pendant laquelle le processus a occupé le processeur depuis son lancement.
- `CMD` : le programme qui a généré ce processus. Si vous voyez plusieurs fois le même programme, c'est que celui-ci s'est dupliqué en plusieurs processus (c'est le cas de MySQL, par exemple).

Dans mon cas, on distingue deux processus : `bash` (qui correspond à l'invite de commandes qui gère les commandes) et `ps` que je viens de lancer.



Deux processus, c'est tout ?

En fait, quand on utilise ps sans argument comme on vient de le faire, il affiche seulement les processus lancés par le même utilisateur (ici « mateo21 ») dans la même console (ici « pts/0 »). Cela limite énormément les processus affichés, car beaucoup sont lancés par root (l'utilisateur administrateur de la machine) et ne sont pas lancés depuis la même console que la vôtre.

La commande ps vous permet d'utiliser énormément d'options. Regardez le manuel pour avoir une petite idée de tout ce que vous pouvez faire avec, vous allez prendre peur.

Plutôt que de faire une longue liste des paramètres possibles, je vous propose quelques combinaisons de paramètres utiles à retenir.

ps -ef : lister tous les processus

Avec ps -ef, vous pouvez avoir la liste de tous les processus lancés par tous les utilisateurs sur toutes les consoles :

Code : Console

```
$ ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 01:01 ?        00:00:01 /sbin/init
root           2     1  0 01:01 ?        00:00:00 [migration/0]
root           3     1  0 01:01 ?        00:00:00 [ksoftirqd/0]
root           4     1  0 01:01 ?        00:00:00 [watchdog/0]
root           5     1  0 01:01 ?        00:00:00 [events/0]
root           6     1  0 01:01 ?        00:00:00 [khelper]
root           7     1  0 01:01 ?        00:00:00 [kthread]
root          30     7  0 01:01 ?        00:00:00 [kblockd/0]
root        2462     1  0 01:01 ?        00:00:00 /sbin/udevd --daemon
root        3292     7  0 01:01 ?        00:00:00 [kpsmoused]
root        3448     7  0 01:01 ?        00:00:00 [kgameportd]
root        4021     1  0 01:02 tty4     00:00:00 /sbin/getty 38400 tty4
root        4022     1  0 01:02 tty5     00:00:00 /sbin/getty 38400 tty5
root        4024     1  0 01:02 tty2     00:00:00 /sbin/getty 38400 tty2
root        4027     1  0 01:02 tty3     00:00:00 /sbin/getty 38400 tty3
root        4030     1  0 01:02 tty1     00:00:00 /sbin/getty 38400 tty1
root        4040     1  0 01:02 tty6     00:00:00 /sbin/getty 38400 tty6
root        4266     1  0 01:02 ?        00:00:00 /usr/sbin/acpid -
c /etc/acpi/eve
root        4363     1  0 01:02 ?        00:00:00 /sbin/syslogd
root        4417     1  0 01:02 ?
        00:00:00 /bin/dd bs 1 if /proc/kmsg of /v
klog        4419     1  0 01:02 ?        00:00:00 /sbin/klogd  -
P /var/run/klogd/km
103         4440     1  0 01:02 ?        00:00:00 /usr/bin/dbus-daemon --
system
107         4456     1  0 01:02 ?        00:00:03 /usr/sbin/hald

...
```

Il y en a vraiment beaucoup, je n'ai pas recopié la liste complète ici.

Vous noterez l'apparition de la colonne UID (*User ID*) qui indique le nom de l'utilisateur qui a lancé la commande. Il y en a beaucoup, lancés par root automatiquement au démarrage de la machine, dont vous n'avez jamais entendu parler.

ps -ejH : afficher les processus en arbre

Cette option intéressante vous permet de regrouper les processus sous forme d'arborescence. Plusieurs processus sont des « enfants » d'autres processus, cela vous permet de savoir qui est à l'origine de quel processus.

Code : Console

```
$ ps -ejH
  PID  PGID   SID TTY          TIME CMD
    1     1     1 ?        00:00:01 init
    2     1     1 ?        00:00:00   migration/0
    3     1     1 ?        00:00:00   ksoftirqd/0
    4     1     1 ?        00:00:00   watchdog/0
    5     1     1 ?        00:00:00   events/0
    6     1     1 ?        00:00:00   khelper
<surligne>    7     1     1 ?        00:00:00   kthread</surligne>
   30     1     1 ?        00:00:00     kblockd/0
   31     1     1 ?        00:00:00     kacpid
   32     1     1 ?        00:00:00     kacpi_notify
   93     1     1 ?        00:00:00     kseriod
  118     1     1 ?        00:00:04     pdflush
  119     1     1 ?        00:00:00     pdflush
  120     1     1 ?        00:00:01     kswapd0
  121     1     1 ?        00:00:00     aio/0
 1930     1     1 ?        00:00:00     ksuspend_usbd
 1931     1     1 ?        00:00:00     khubd
 2061     1     1 ?        00:00:00     ata/0
 2062     1     1 ?        00:00:00     ata_aux
 2094     1     1 ?        00:00:00     scsi_eh_0
 2263     1     1 ?        00:00:09     kjournald
 3292     1     1 ?        00:00:00     kpsmoused
 3448     1     1 ?        00:00:00     kgameportd
 4521  4521  4521 ?        00:00:00   NetworkManager
 4538  4538  4538 ?        00:00:01   avahi-daemon
 4539  4539  4539 ?        00:00:00     avahi-daemon
 4556  4556  4556 ?        00:00:00   NetworkManagerD
 4569  4569  4569 ?        00:00:00   system-tools-ba
 4570  4569  4569 ?        00:00:00     dbus-daemon
 4593  4593  4593 ?        00:00:00   gdm
 4594  4594  4593 ?        00:00:00     gdm
 4625  4625  4625 tty7     00:05:56       Xorg
 5012  5012  5012 ?        00:00:01       gnome-session
 5057  5057  5057 ?        00:00:00         ssh-agent
 5080  5012  5012 ?        00:00:25         metacity
 5083  5012  5012 ?        00:00:16         gnome-panel
 5089  5012  5012 ?        00:00:31         nautilus
 5098  5012  5012 ?        00:00:01         update-notifier
 5102  5012  5012 ?        00:00:01         evolution-alarm
 5107  5012  5012 ?        00:00:02         nm-applet
 5112  5012  5012 ?        00:01:18         gnome-cups-icon
 4640  4640  4640 ?        00:00:05   cupsd
 4672  4672  4672 ?        00:00:00   hpiod
```

Dans cette liste, vous pouvez voir que `kthread` (ici surligné) a lancé lui-même de nombreux processus, comme `kacpid`, `pdflush`...
Autre exemple : `gdm` (Gnome Desktop Manager) lance `Xorg` ainsi que gnome-session qui lui-même lance `nautilus`, `gnome-panel`, etc.

`ps -u UTILISATEUR` : lister les processus lancés par un utilisateur

Pour filtrer un peu cette longue liste, on peut utiliser `-u` afin d'obtenir par exemple uniquement les processus que l'on a lancés nous-mêmes.

Code : Console

```
$ ps -u mateo21
  PID TTY          TIME CMD
 5012 ?        00:00:01 gnome-session
 5057 ?        00:00:00 ssh-agent
 5060 ?        00:00:00 dbus-launch
 5061 ?        00:00:00 dbus-daemon
```

```
  5063 ?        00:00:03 gconfd-2
  5066 ?        00:00:00 gnome-keyring-d
  5068 ?        00:00:03 gnome-settings-
  5075 ?        00:00:00 sh
  5076 ?        00:00:00 esd
  5080 ?        00:00:25 metacity
  5083 ?        00:00:16 gnome-panel
  5089 ?        00:00:31 nautilus
```

Ici, j'obtiens uniquement les processus lancés par l'utilisateur « mateo21 », ce qui filtre déjà pas mal les autres processus système lancés par root.

`top` : liste dynamique des processus

La liste donnée par `ps` a un défaut : elle est **statique** (elle ne bouge pas). Or, votre ordinateur, lui, est en perpétuel mouvement. De nombreux processus apparaissent et disparaissent régulièrement.

Comment avoir une liste régulièrement mise à jour ? Avec la commande `top` !

Essayez-la :

Code : Console

```
top - 13:31:30 up 12:30,  3 users,  load average: 0.01, 0.07, 0.11
Tasks:  96 total,   3 running,  93 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.8%us,  0.6%sy,  0.0%ni, 97.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    515984k total,   453652k used,    62332k free,    69036k buffers
Swap:   240932k total,    31496k used,   209436k free,   246404k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4625 root      15   0 38572  14m 6676 R  1.2  2.9   6:01.00 Xorg
 5068 mateo21   15   0 29760 9.8m 8008 S  0.6  1.9   0:03.69 gnome-settings-
 5112 mateo21   15   0 48612 8440 6844 S  0.6  1.6   1:19.45 gnome-cups-icon
    1 root      18   0  2908 1848  524 S  0.0  0.4   0:01.50 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.01 ksoftirqd/0
    4 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      10  -5     0    0    0 S  0.0  0.0   0:00.66 events/0
    6 root      10  -5     0    0    0 S  0.0  0.0   0:00.02 khelper
    7 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
   30 root      10  -5     0    0    0 S  0.0  0.0   0:00.55 kblockd/0
   31 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   32 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
   93 root      10  -5     0    0    0 S  0.0  0.0   0:00.02 kseriod
  118 root      15   0     0    0    0 S  0.0  0.0   0:04.84 pdflush
  119 root      15   0     0    0    0 S  0.0  0.0   0:00.20 pdflush
  120 root      10  -5     0    0    0 S  0.0  0.0   0:01.29 kswapd0
```

Cette liste est **interactive** et régulièrement mise à jour.

En haut, vous retrouvez l'uptime et la charge, mais aussi la quantité de processeur et de mémoire utilisée. Nous n'entrerons pas dans les détails à ce niveau car cela demanderait un peu trop d'explications avancées sur le fonctionnement du système d'exploitation. Néanmoins, si vous savez lire la charge et la mémoire disponible, vous pouvez déjà vous faire une idée de ce qui se passe.

En dessous, vous avez la liste des processus.

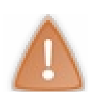

Pourquoi y a-t-il si peu de processus ?

`top` ne peut pas afficher tous les processus à la fois, il ne conserve que les premiers pour qu'ils tiennent sur une « page » de la console.

Par défaut, les processus sont triés par taux d'utilisation du processeur (colonne `%CPU`). Les processus que vous voyez tout en haut de cette liste sont donc actuellement les plus gourmands en processeur. Ce sont peut-être eux que vous devriez cibler en premier si vous sentez que votre système est surchargé.

On navigue à l'intérieur de ce programme en appuyant sur certaines touches du clavier. En voilà au moins deux à connaître :

- **q** : ferme `top` ;
- **h** : affiche l'aide, et donc la liste des touches utilisables.



Attention à la différence entre majuscules et minuscules ! Taper « h » n'a pas le même effet que de taper « H » !

Mis à part cela, voici quelques commandes à connaître au sein de `top` qui peuvent vous être utiles.

- **B** : met en gras certains éléments.
- **f** : ajoute ou supprime des colonnes dans la liste.
- **F** : change la colonne selon laquelle les processus sont triés. En général, laisser le tri par défaut en fonction de `%CPU` est suffisant.
- **u** : filtre en fonction de l'utilisateur que vous voulez.
- **k** : tue un processus, c'est-à-dire arrête ce processus. Ne vous inquiétez pas, en général les processus ne souffrent pas. On vous demandera le numéro (`PID`) du processus que vous voulez tuer. Nous reviendrons sur l'arrêt des processus un peu plus loin.
- **s** : change l'intervalle de temps entre chaque rafraîchissement de la liste (par défaut, c'est toutes les trois secondes).

Vous voilà parés à utiliser `top` ! ;-)
Je l'utilise principalement pour voir la charge évoluer régulièrement tout en surveillant les processus les plus gourmands qui peuvent poser un problème.

Ctrl + C & kill : arrêter un processus

Parfois, rien ne va plus. Un processus s'emballe et ne veut pas s'arrêter. Cela arrive partout, même sous Linux. À la différence de Windows toutefois, vous ne devriez pas avoir le réflexe de redémarrer « pour que ça aille mieux ». Tout peut être résolu en arrêtant les processus qui vous gênent et en les relançant au besoin.

Il y a plusieurs façons d'arrêter un processus, nous allons les étudier ici.

`Ctrl + C` : arrêter un processus lancé en console

La combinaison de touches `Ctrl + C` est à connaître. Cela demande (gentiment) l'arrêt du programme console en cours d'exécution à l'écran. Ce raccourci se comporte ainsi en mode console seulement. En effet, en mode graphique, le comportement est le même que sous Windows : cela permet d'effectuer une copie dans le presse-papier. Notez que pour copier-coller sous Linux, on utilise souvent une autre technique : on sélectionne du texte et on clique avec la molette de la souris pour le coller ailleurs.

Prenez une commande qui n'en finit plus, comme par exemple un `find` sur l'ensemble du disque. Celui-ci va analyser tout votre disque dur à la recherche du fichier demandé. Si vous trouvez cela trop long et que vous voulez arrêter le programme en cours de route, il vous suffit de taper `Ctrl + C` :

Code : Console

```
# find / -name "*log*"
/dev/log
/bin/login
/sys/module/scsi_mod/parameters/scsi_logging_level
/sys/module/ehci_hcd/parameters/log2_irq_thresh
```

La liste aurait dû être beaucoup plus longue. Mais j'ai demandé l'arrêt du programme avec `Ctrl + C`, ce qui fait que j'ai pu « retrouver » l'invite de commandes rapidement et facilement.

Taper `Ctrl + C` ne coupe pas le programme brutalement, cela lui demande gentiment de s'arrêter, comme si vous aviez cliqué sur la croix pour fermer une fenêtre.

`kill` : tuer un processus

`Ctrl + C` ne fonctionne que sur un programme actuellement ouvert dans la console. De nombreux programmes tournent pourtant en arrière-plan, et `Ctrl + C` n'aura aucun effet sur eux.

C'est là que vous devez utiliser `kill` si vous voulez les arrêter (on dit aussi « tuer », c'est pareil même si ça a l'air violent).

Pour vous en servir, il faudra auparavant récupérer le `PID` du ou des processus que vous voulez tuer. Pour cela, deux solutions :

- `ps` ;
- `top`.

Ces deux commandes que nous venons de voir vous indiquent le `PID` (numéro d'identification) de chaque processus. Par exemple avec `ps` :

Code : Console

```
$ ps -u mateo21
  PID TTY          TIME CMD
 5012 ?        00:00:01 gnome-session
 5057 ?        00:00:00 ssh-agent
 5060 ?        00:00:00 dbus-launch
 5061 ?        00:00:00 dbus-daemon
 5063 ?        00:00:03 gconfd-2
 5066 ?        00:00:00 gnome-keyring-d
 5068 ?        00:00:03 gnome-settings-
 5075 ?        00:00:00 sh
 5076 ?        00:00:00 esd
 5080 ?        00:00:26 metacity
 5083 ?        00:00:17 gnome-panel

...

25227 pts/1    00:00:00 bash
32617 pts/1    00:00:00 man
32627 pts/1    00:00:00 pager
32703 pts/0    00:00:00 ps
```

Supposons qu'on souhaite arrêter Firefox. On peut filtrer cette longue liste avec `grep` et un pipe que nous avons appris à utiliser.

Code : Console

```
$ ps -u mateo21 | grep firefox
32678 ?        00:00:03 firefox-bin
```

Hop là, on a filtré Firefox de cette longue liste et on a même récupéré son `PID`. Il ne nous reste plus qu'à le tuer, avec la commande suivante :

Code : Console

```
kill 32678
```

Si tout va bien, la commande ne renvoie rien. Sinon, une erreur devrait s'afficher dans la console.

Vous pouvez aussi tuer plusieurs processus d'un seul coup en indiquant plusieurs `PID` à la suite :

Code : Console

```
kill 32678 2768 33071
```



Attention : même si `kill` est par défaut une commande « gentille » qui demande simplement au processus de s'arrêter, évitez de tuer des processus que vous ne connaissez pas. Beaucoup d'entre eux sont essentiels au bon fonctionnement de votre système, surtout ceux qui ont été lancés par root.



J'ai essayé, mais Firefox a l'air vraiment complètement planté et il refuse de s'arrêter. Il n'y a pas moyen d'être un peu plus… direct ?

Vous voulez tuer un processus sans lui laisser le choix ?
C'est tout à fait possible, mais à n'utiliser que dans le cas d'un programme complètement planté que vous voulez vraiment arrêter !

Avec `kill -9` (comme le chiffre 9, oui, oui), vous demandez à Linux de tuer le processus sans lui laisser le temps de s'arrêter proprement. Cela peut faire le ménage quand rien ne va plus.

Code : Console

```
kill -9 32678
```

… tuera le processus n°32678 (Firefox, dans mon cas) immédiatement sans lui laisser le temps de finir.

`killall` : tuer plusieurs processus

Souvenez-vous : je vous ai dit que certains programmes se dupliquaient en plusieurs processus. Si vous voulez arrêter l'ensemble de ces processus, comment faire ? Heureusement, vous avez des armes pour éradiquer cette vermine.

Vous pourriez, certes, tuer tous les processus en récupérant un à un leur `PID`. Mais il y a plus rapide : `killall` (« tuez-les tous ! »).

Contrairement à `kill`, `killall` attend le nom du processus à tuer et non son `PID`.

Supposons que nous ayons trois processus `find` en cours d'exécution que nous souhaitions arrêter.

Code : Console

```
$ ps -u mateo21 | grep find
  675 pts/1    00:00:01 find
  678 pts/2    00:00:00 find
  679 pts/3    00:00:01 find
```

Pour tous les tuer, il faudra donc taper :

Code : Console

```
$ killall find
```

Si la commande ne renvoie rien, c'est que tout s'est bien passé.

En revanche, si vous avez :

Code : Console

```
$ killall find
find: aucun processus tué
```

… cela signifie qu'il n'y avait aucun processus de ce nom à tuer. Soit le processus n'est plus là, soit vous n'avez pas écrit correctement son nom. Vérifiez ce nom à nouveau avec la commande `ps`.

halt & reboot : arrêter et redémarrer l'ordinateur

Nous venons d'apprendre à arrêter des processus avec `kill`. Je pense que le moment est bien choisi pour découvrir comment arrêter et redémarrer l'ordinateur.

Comme je vous le disais plus tôt, il est assez rare que l'on soit forcé d'arrêter ou de redémarrer l'ordinateur. À moins d'avoir mis à jour le kernel (noyau) de Linux, il n'est jamais nécessaire de redémarrer.
L'arrêt et le redémarrage d'un serveur sous Linux sont réellement des opérations exceptionnelles.



Mais j'ai installé Linux sur mon ordinateur personnel ! Je n'en fais pas un serveur. J'ai le droit de l'arrêter ou de le redémarrer quand même, non ?

En effet, et je suppose que vous n'avez pas attendu ce chapitre pour le faire. ;-)
Vous pouviez arrêter et redémarrer l'ordinateur via l'interface graphique (Unity, KDE, …). Mais en console, savez-vous le faire ?

`halt` : arrêter l'ordinateur

La commande `halt` commande l'arrêt immédiat de l'ordinateur. Il faut être root pour arrêter la machine ; vous devrez donc taper :

Code : Console

```
$ sudo halt
```

Un message sera affiché dans la console pour annoncer l'arrêt de l'ordinateur.

`reboot` : redémarrer l'ordinateur

De même, il existe la commande `reboot` pour redémarrer l'ordinateur. Il faut à nouveau être root :

Code : Console

```
$ sudo reboot
```

Le redémarrage prend effet immédiatement.



Les commandes `halt` et `reboot` appellent en réalité la commande `shutdown` avec des paramètres spécifiques. N'hésitez pas à lire sa page du manuel, vous verrez que vous pouvez par exemple programmer un arrêt ou un redémarrage à une heure précise ou au bout d'un certain temps.

En résumé

- Linux est multi-tâches (plusieurs programmes peuvent tourner en même temps) et multi-utilisateurs (plusieurs utilisateurs peuvent se servir de la même machine en même temps en s'y connectant via Internet).
- `w` indique quels utilisateurs sont sur la machine, ce qu'ils font et quelques autres statistiques comme la charge de travail de la machine et son uptime.
- `ps` affiche la liste des processus, c'est-à-dire des programmes qui tournent sur la machine. `top` est un équivalent qui met à jour automatiquement la liste au fil du temps.
- La combinaison de touches `Ctrl + C` permet d'arrêter une commande en cours d'exécution dans la console afin de pouvoir reprendre la main.
- `kill` tue un processus, ce qui signifie qu'il lui demande de s'arrêter. Il a besoin du numéro du processus, généralement fourni par `ps` ou `top`. Si le processus ne s'arrête pas, on peut utiliser le paramètre `-9` qui coupe brutalement le processus (avec risque de perte de données).
- `halt` commande l'arrêt de l'ordinateur, `reboot` son redémarrage.

 Exécuter des programmes en arrière-plan

Nousavons commencé à découvrir ce qu'étaient les processus dans le chapitre précédent. Nous savons désormais comment les lister, les trier, les filtrer et enfin comment les tuer.

Ici, je vous propose d'aller plus loin et de découvrir **l'exécution en arrière-plan**. A priori, la console a quelque chose de frustrant : on a l'impression qu'on ne peut lancer qu'un seul programme à la fois par console. Or, c'est tout à fait faux ! … Mais encore faut-il savoir comment faire tourner des programmes en arrière-plan.

Il existe un certain nombre de techniques plus ou moins sophistiquées. Il est recommandé de les connaître car, parfois, on souhaite tout faire au sein d'une seule et même console.

"&" & nohup : lancer un processus en arrière-plan

Lorsque vous vous apprêtez à lancer une opération un peu longue, comme une grosse copie de fichiers par exemple, vous n'avez peut-être pas envie de patienter sagement le temps que la commande s'exécute pour pouvoir faire autre chose en attendant. Certes, on peut ouvrir une autre console me direz-vous. Il y a des cas cependant où l'on n'a accès qu'à une seule console, ou encore tout simplement pas envie d'en ouvrir une autre (la flemme, vous connaissez ? ;-).

Contrairement aux apparences, plusieurs programmes peuvent tourner en même temps au sein d'une même console. Ce n'est pas parce qu'on ne peut pas afficher plusieurs fenêtres comme dans un environnement graphique qu'on est bloqué sur un seul programme à la fois ! Encore faut-il connaître les techniques qui permettent de lancer une commande en tâche de fond…

`&` : lancer un processus en arrière-plan

La première technique que je veux vous faire découvrir est très simple : elle consiste à rajouter le petit symbole `&` à la fin de la commande que vous voulez envoyer en arrière-plan.



Le symbole `&` s'appelle le « et commercial » ou encore l'« esperluette ». Il est présent sur la touche 1 d'un clavier AZERTY.

Prenons par exemple la commande `cp` qui permet de copier des fichiers.

Je vous propose de copier un gros fichier vidéo (ce qui prend en général du temps), comme ceci :

Code : Console

```
$ cp video.avi copie_video.avi &
[1] 16504
```



Notez que l'espace avant le `&` à la fin n'est pas obligatoire.

On vous renvoie deux informations.

- `[1]` : c'est le numéro du processus en arrière-plan dans cette console. Comme c'est le premier processus que nous envoyons en arrière-plan, il prend le numéro 1.
- `16504` : c'est le numéro d'identification général du processus (le fameux `PID` dont on a déjà parlé). Cette information vous permet de tuer le processus avec `kill` si nécessaire.

Maintenant, vous ne voyez peut-être rien, mais le processus est bel et bien en train de tourner en « tâche de fond ».

Si vous essayez de faire la même chose avec d'autres commandes, par exemple sur un `find`, vous risquez d'être surpris : les messages renvoyés par la commande s'affichent toujours dans la console ! Vous pouvez certes écrire du texte et lancer d'autres commandes pendant ce temps (essayez), mais c'est un peu frustrant de voir ces messages apparaître dans la console !

Heureusement, vous savez maintenant rediriger la sortie pour ne pas être importunés :

Code : Console

```
$ find / -name "*log" > sortiefind &
[1] 18191
```

Les résultats seront maintenant écrits dans le fichier `sortiefind` au lieu d'être affichés dans la console. De plus, la commande s'exécute en fond et ne nous importune plus.

Notez que pour être sûrs de ne pas être dérangés du tout, vous devrez aussi rediriger les erreurs (par exemple avec `2>&1`), ce qui peut nous donner une jolie commande comme celle-ci :

Code : Console

```
$ find / -name "*log" > sortiefind 2>&1 &
[1] 18231
```

Il reste toutefois un problème : le processus est « attaché » à votre console. Si vous fermez la console sur laquelle vous êtes, le processus sera tué et ne s'exécutera donc pas jusqu'au bout.

`nohup` : détacher le processus de la console

L'option `&`, bien qu'assez couramment utilisée, a ce défaut non négligeable : le processus reste attaché à la console, ce qui veut dire que si la console est fermée ou que l'utilisateur se déconnecte, le processus sera automatiquement arrêté.

Si on veut que le processus continue, il faut lancer la commande via `nohup`. Cela s'utilise comme ceci :

Code : Console

```
nohup commande
```

Par exemple, voici ce que ça donne si on lance la copie via un `nohup` :

Code : Console

```
$ nohup cp video.avi copie_video.avi
nohup: ajout à la sortie de `nohup.out'
```

La sortie de la commande est par défaut redirigée vers un fichier `nohup.out`. Aucun message ne risque donc d'apparaître dans la console.

D'autre part, la commande est maintenant immunisée contre la fermeture de la console. Elle continuera de fonctionner quoi qu'il arrive (sauf si on lui envoie un `kill`, bien sûr).



`nohup` est très utile par exemple lorsque vous vous connectez à un serveur.
Imaginons que vous voulez lancer un programme (comme un serveur de jeu) : celui-ci s'arrêtera de fonctionner dès que vous vous serez déconnectés de la ligne de commandes du serveur. Vous n'allez pas rester connectés juste pour que le programme continue à fonctionner ! Heureusement, `nohup` vous préserve de ce problème.

Ctrl + Z, jobs, bg & fg : passer un processus en arrière-plan

Voyons maintenant le problème différemment : vous avez lancé la commande sans penser à rajouter un petit `&` à la fin. Malheureusement, la commande prend beaucoup plus de temps à s'exécuter que ce que vous aviez prévu. Êtes-vous condamnés à attendre qu'elle soit terminée pour reprendre la main sur l'invite de commandes ? Bien sûr que non !

Il y a une série de commandes et de raccourcis qu'il vous faut absolument connaître ! Nous allons les étudier un par un dès

maintenant.

`Ctrl + Z` : mettre en pause l'exécution du programme

Reprenons le cas de notre grosse copie de fichiers. Cette fois, je suppose que vous l'avez lancée sans le petit symbole `&` :

Code : Console

```
$ cp video.avi video_copie.avi
```

Si vous n'avez pas de gros fichier sous la main pour faire le test, vous pouvez aussi faire un `top`.

Tapez maintenant `Ctrl + Z` pendant l'exécution du programme. Celui-ci va s'arrêter et vous allez immédiatement reprendre la main sur l'invite de commandes.

Code : Console

```
[1]+  Stopped                 top
mateo21@mateo21-desktop:~$
```

Vous noterez que nous avons plusieurs informations : le numéro du processus en arrière-plan (ici `[1]`), son état (`Stopped`) et le nom de la commande qui a lancé ce processus.

Le processus est maintenant dans un état de pause. Il ne s'exécute pas mais reste en mémoire.

`bg` : passer le processus en arrière-plan (*background*)

Maintenant que le processus est en « pause » et qu'on a récupéré l'invite de commandes, tapez :

Code : Console

```
$ bg
[1]+ top &
```

C'est tout, pas besoin de paramètre.

Qu'est-ce que cela fait ? Cela commande la reprise du processus, mais cette fois en arrière-plan. Il continuera à s'exécuter à nouveau, mais en tâche de fond.

En résumé, si vous avez lancé une commande par erreur en avant-plan et que vous voulez récupérer l'invite de commandes, il faudra faire dans l'ordre :

- `Ctrl + Z` : pour mettre en pause le programme et récupérer l'invite de commandes ;
- `bg` : pour que le processus continue à tourner mais en arrière-plan.

`jobs` : connaître les processus qui tournent en arrière-plan

Vous pouvez envoyer autant de processus en arrière-plan que vous voulez au sein d'une même console :

- soit en les lançant directement en arrière-plan avec un `&` à la fin de la commande ;
- soit en utilisant la technique du `Ctrl + Z` suivi de `bg` que vous venez d'apprendre.

Comment savoir maintenant quels sont les processus qui tournent en arrière-plan ? Vous pourriez, certes, recourir à la commande

`ps`, mais celle-ci vous donnera tous les processus. C'est un peu trop.

Heureusement, il existe une commande qui liste uniquement les processus qui tournent en fond au sein d'une même console : `jobs`.

Code : Console

```
$ jobs
[1]-  Stopped                 top
[2]+  Stopped                 find / -name "*log*" > sortiefind 2>&1
```

Encore une fois, vous avez le numéro du processus qui tourne en fond (à ne pas confondre avec le `PID`), son état et son nom.

`fg` : reprendre un processus au premier plan (*foreground*)

La commande `fg` renvoie un processus au premier plan.

Code : Console

```
$ fg
```

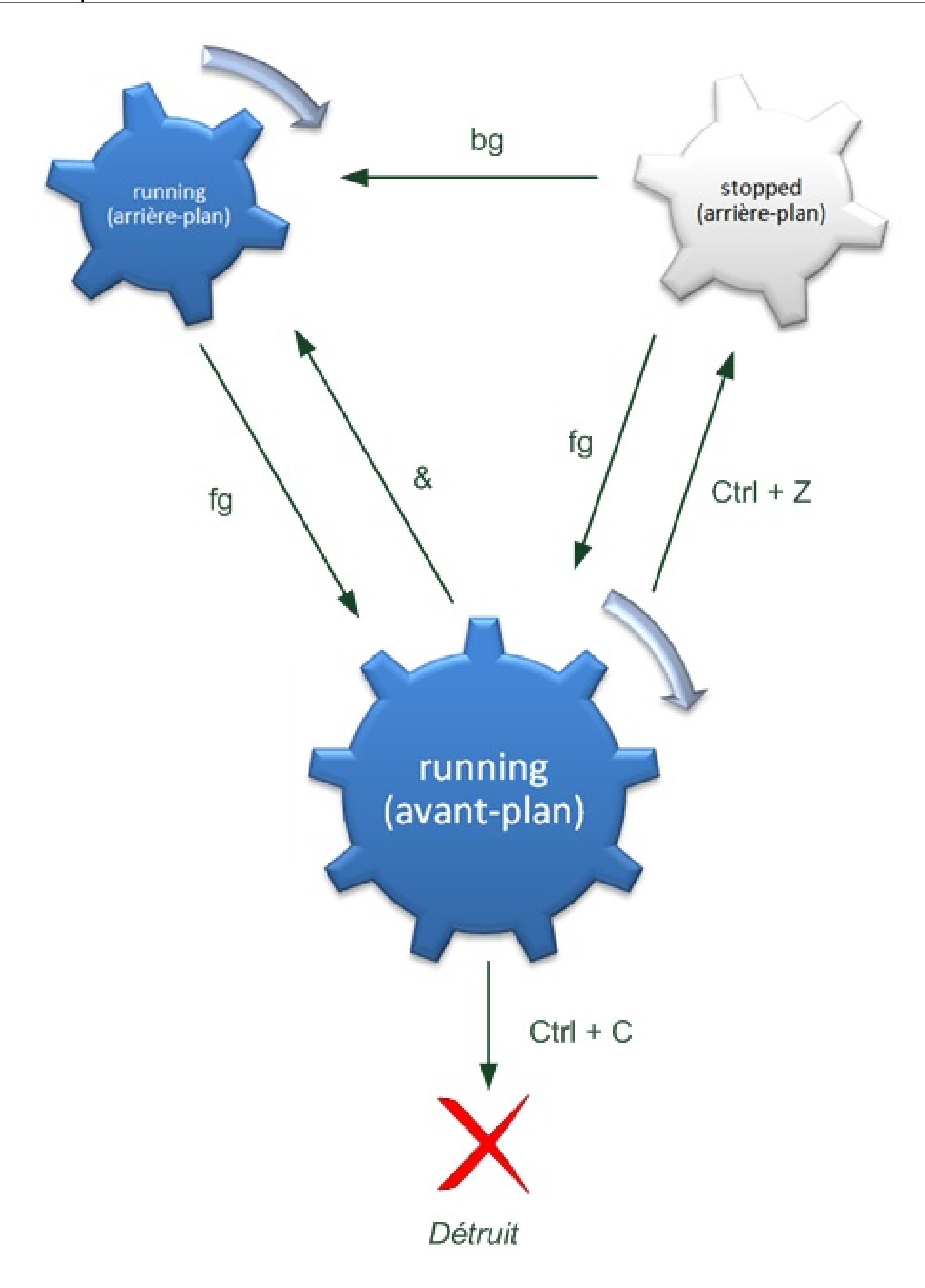
Si vous avez un seul processus listé dans les `jobs`, c'est ce processus qui sera remis au premier plan.
Si, comme moi tout à l'heure, vous avez plusieurs processus en arrière-plan, il faudra préciser lequel vous voulez récupérer. Par exemple, voici comment reprendre le `find` qui était le `job` n° 2 :

Code : Console

```
$ fg %2
```

Résumé des états possibles des processus

Je pense qu'un schéma s'impose maintenant. Dans la figure suivante, je résume tout ce que nous avons vu jusqu'ici, à l'exception de `nohup` qui est une commande un peu à part.



Expliquons un peu ce schéma !

Par défaut, un processus est lancé dans l'état `running` à l'avant-plan. On peut l'arrêter avec la combinaison `Ctrl + C`, auquel cas il sera détruit.

Mais on peut aussi l'envoyer en arrière-plan. Si on l'exécute dès le départ avec un `&`, il sera à l'état `running` à l'arrière-plan. Si on choisit de faire `Ctrl + Z`, il passera à l'état `Stopped` à l'arrière-plan. Il faudra taper `bg` pour le faire passer à nouveau à l'état `running` en arrière-plan.

Enfin, la commande `fg` renvoie un processus de l'arrière-plan vers l'avant-plan.

Prenez cinq minutes pour bien analyser ce schéma et vérifier que vous avez compris l'essentiel de ce chapitre, c'est vraiment important. Il résume à peu près tout ce qu'il faut savoir. Il manque seulement `nohup` que j'ai mis à part comme je vous l'ai dit.

screen : plusieurs consoles en une

Il nous reste à découvrir une commande un peu particulière que j'ai volontairement réservée pour la fin : `screen`.
Pourquoi ai-je attendu avant d'en parler ? Tout simplement parce que, contrairement à ce que nous avons vu jusqu'ici, ce n'est pas une commande « standard » qui est installée par défaut sur toutes les distributions Linux. Parfois, vous n'aurez pas accès à `screen` (parce que vous n'êtes pas root sur la machine) et il faudra vous débrouiller avec les commandes que l'on vient de voir.

Si toutefois vous êtes les maîtres de la machine (ce qui est votre cas si vous avez installé Linux chez vous), je peux vous

recommander d'installer le programme `screen`.

Code : Console

```
$ sudo apt-get install screen
```

De quoi s'agit-il ?
`screen` est un multiplicateur de terminal. Derrière ce nom un peu pompeux qui peut faire peur – je le reconnais – se cache en fait un programme capable de gérer plusieurs consoles au sein d'une seule, un peu comme si chaque console était une fenêtre !



Imaginez que `screen` est un programme qui permet entre autres de faire une **mise en veille prolongée** de votre console, tout comme vous le faites peut-être avec votre ordinateur portable qui se retrouve exactement dans l'état où vous l'avez laissé en l'éteignant.
Concrètement, j'ai souvent tendance à utiliser `screen` sur un serveur. Cela me permet par exemple de lancer un serveur de jeu dans une console `screen`, de quitter le serveur, puis de revenir l'administrer plus tard au besoin en récupérant la console dans l'état où je l'ai laissée.

Lorsque vous avez installé `screen`, essayez-le en tapant tout simplement :

Code : Console

```
$ screen
```

Un message s'affiche, précisant tout d'abord que le programme est un logiciel libre. Il indique ensuite l'adresse e-mail de l'auteur à laquelle on peut envoyer, je cite « des t-shirts, de l'argent, de la bière et des pizzas ». Bon… passons. :-)

Code : Console

```
Screen version 4.00.03 (FAU) 23-Oct-06

Copyright (c) 1993-2002 Juergen Weigert, Michael Schroeder
Copyright (c) 1987 Oliver Laumann

This program is free software; you can redistribute it and/or modify it under
the terms of the GNU General Public License as published by the Free Software
Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT
ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with
this program (see the file COPYING); if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.

Send bugreports, fixes, enhancements, t-shirts, money, beer & pizza to
screen@uni-erlangen.de

                        [Press Space or Return to end.]
```

Tapez `Entrée` ou `Espace` pour passer ce message.

À première vue, il ne se passe rien de bien extraordinaire : on retrouve une console vide. Mais mine de rien, nous nous trouvons dans une console « émulée », non pas dans la « vraie » console où nous étions tout à l'heure. Vous pouvez en sortir en tapant `Ctrl + D` ou `exit`, comme si vous quittiez une console normalement.
Vous retrouverez alors votre console habituelle où vous avez lancé `screen` :

Code : Console

```
mateo21@mateo21-desktop:~$ screen
[screen is terminating]
```

Bon, maintenant que vous savez sortir de `screen`, retournez-y. :-)

Il faut savoir que sous `screen`, tout se fait à partir de combinaisons de touches de la forme suivante : `Ctrl + a + autre touche`. En fait, vous devez taper `Ctrl + a`, relâcher ces touches (lever les mains du clavier) et ensuite appuyer sur une autre touche.

`Ctrl + a` puis `?` : afficher l'aide

Essayez de taper `Ctrl + a`, puis tapez `?`. L'aide devrait alors s'afficher :

Code : Console

```
Screen key bindings, page 1 of 2.

                       Command key:  ^A   Literal ^A:  a

  break        ^B b         license     ,            removebuf    =
  clear        C            lockscreen  ^X x         reset        Z
  colon        :            log         H            screen       ^C c
  copy         ^[ [         login       L            select       '
  detach       ^D d         meta        a            silence      _
  digraph      ^V           monitor     M            split        S
  displays     *            next        ^@ ^N sp n   suspend      ^Z z
  dumptermcap  .            number      N            time         ^T t
  fit          F            only        Q            title        A
  flow         ^F f         other       ^A           vbell        ^G
  focus        ^I           pow_break   B            version      v
  hardcopy     h            pow_detach  D            width        W
  help         ?            prev        ^H ^P p ^?   windows      ^W w
  history      { }          quit        \            wrap         ^R r
  info         i            readbuf     <            writebuf     >
  kill         K k          redisplay   ^L l         xoff         ^S s
  lastmsg      ^M m         remove      X            xon          ^Q q

                  [Press Space for next page; Return to end.]
```

Il y a deux pages de commandes. Avec `Espace` vous allez à la page suivante ; avec `Entrée`, vous refermez l'aide.

Comment lire cette page d'aide ? Par exemple, si vous voulez connaître la version du programme (milieu de la troisième colonne), il faudra taper `Ctrl + a` suivi de `v` (la lettre minuscule). Toutes les touches que vous voyez là doivent impérativement être précédées d'un `Ctrl + a`.
Notez par ailleurs que l'accent circonflexe ^ signifie ici `Ctrl`.

Les principales commandes de `screen`

Je ne connais pas toutes ces commandes, mais je vais vous en présenter les principales, celles qui selon moi peuvent vous être utiles.

- `Ctrl + a` puis `c` : créer une nouvelle « fenêtre ».
- `Ctrl + a` puis `w` : afficher la liste des « fenêtres » actuellement ouvertes. En bas de l'écran vous verrez par exemple apparaître : `0-$ bash  1*$ bash`. Cela signifie que vous avez deux fenêtres ouvertes, l'une numérotée 0, l'autre 1. Celle sur laquelle vous vous trouvez actuellement contient une étoile `*` (on se trouve donc ici dans la fenêtre n° 1).

- `Ctrl + a` puis `A` : renommer la fenêtre actuelle. Ce nom apparaît lorsque vous affichez la liste des fenêtres avec `Ctrl + a` puis `w`.
- `Ctrl + a` puis `n` : passer à la fenêtre suivante (*next*).
- `Ctrl + a` puis `p` : passer à la fenêtre précédente (*previous*).
- `Ctrl + a` puis `Ctrl + a` : revenir à la dernière fenêtre utilisée.
- `Ctrl + a` puis un chiffre de 0 à 9 : passer à la fenêtre n° X.
- `Ctrl + a` puis `"` : choisir la fenêtre dans laquelle on veut aller.
- `Ctrl + a` puis `k` : fermer la fenêtre actuelle (*kill*).



`screen` est sensible à la casse pour les commandes ! Faites donc bien la différence entre « c » et « C » par exemple.

Il nous reste deux options très intéressantes de `screen` à découvrir et qui méritent une attention particulière : `split` et `detach`.

`Ctrl + a` puis `S` : découper `screen` en plusieurs parties (*split*)

`Ctrl + a` puis `S` coupe l'écran en deux pour afficher deux consoles à la fois (*split*). Il est possible de répéter l'opération plusieurs fois pour couper en trois, quatre, ou plus (dans la mesure du possible, parce qu'après les consoles sont toutes petites).

Voici, en figure suivante, ce que vous voyez après avoir *splitté* l'écran une fois.

```
mateo21@mateo21-desktop:~$ ls
bin              Desktop              ies4linux-latest.tar.gz  tuto
copie_video.avi  Examples             Images                   video.avi
cours_unix.txt   ies4linux-2.99.0.1   sortiefind               video_copie.avi
mateo21@mateo21-desktop:~$ rm copie_video.avi
mateo21@mateo21-desktop:~$ 



 0 bash









 --
```

L'écran est bien découpé en deux, mais la « fenêtre » du bas est vide. Il n'y a même pas d'invite de commandes.

Pour passer d'une fenêtre à une autre, faites `Ctrl + a` puis `Tab`.
Une fois le curseur placé dans la fenêtre du bas, vous pouvez soit créer une nouvelle fenêtre (`Ctrl + a` puis `c`) soit appeler une autre fenêtre que vous avez déjà ouverte (avec `Ctrl + a` puis un chiffre, par exemple).

Vous pourrez, comme dans la figure suivante, afficher par exemple `top` pendant que vous faites des opérations sur la fenêtre du dessus.

```
mateo21@mateo21-desktop:~$ ls
bin               Desktop            ies4linux-latest.tar.gz  tuto
copie_video.avi   Examples           Images                   video.avi
cours_unix.txt    ies4linux-2.99.0.1 sortiefind               video_copie.avi
mateo21@mateo21-desktop:~$ rm copie_video.avi
mateo21@mateo21-desktop:~$
```

```
  0 bash
top - 21:34:26 up  3:39,  3 users,  load average: 0.15, 0.18, 0.12
Tasks:  99 total,   1 running,  97 sleeping,   1 stopped,   0 zombie
Cpu(s):  0.7%us,  1.0%sy,  0.0%ni, 98.0%id,  0.0%wa,  0.3%hi,  0.0%si,  0.0%st
Mem:    515984k total,   480984k used,    35000k free,   111412k buffers
Swap:   240932k total,    33820k used,   207112k free,   197040k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5910 root      15   0  2444  984  784 S  0.7  0.2   0:17.45 vmware-guestd
 5680 root      15   0 41216  16m 6440 S  0.3  3.2   1:16.42 Xorg
20582 mateo21   15   0  2320 1144  880 R  0.3  0.2   0:00.05 top
    1 root      20   0  2912 1844  524 S  0.0  0.4   0:00.97 init
  1 bash
```

La classe de geek, quoi. ;-)

Ah, et pour fermer une fenêtre que vous avez *splittée*, il faudra taper `Ctrl + a` puis `X`. Voilà, vous savez l'essentiel !

`Ctrl + a` puis `d` : détacher `screen`

`Ctrl + a` puis `d` détache `screen` et vous permet de retrouver l'invite de commandes « normale » sans arrêter `screen`. C'est peut-être une des fonctionnalités les plus utiles que nous devons approfondir, et cela nous ramène d'ailleurs à l'exécution de programmes en arrière-plan dont nous avons parlé au début du chapitre.

Concrètement, si vous détachez `screen`, vous retrouvez l'invite de commandes classique :

Code : Console

```
mateo21@mateo21-desktop:~$ screen
[detached]
mateo21@mateo21-desktop:~$
```

L'information `[detached]` apparaît pour signaler que `screen` tourne toujours et qu'il est détaché de la console actuelle. Il continuera donc à tourner quoi qu'il arrive, même si vous fermez la console dans laquelle vous vous trouvez.



Ah, alors c'est comme `nohup` finalement, non ?

En effet, `screen` se comporte comme un `nohup`. La différence est qu'une session `screen` vous permet d'ouvrir plusieurs fenêtres de console à la fois, contrairement à `nohup` qui ne peut lancer qu'un programme à la fois.

Vous pouvez donc partir, quitter la console et revenir récupérer votre session `screen` plus tard. Il faudra simplement taper :

Code : Console

```
$ screen -r
```

… pour retrouver votre session `screen` dans l'état où vous l'avez laissée.

Notez qu'il est possible de faire tourner plusieurs sessions `screen` en fond à la fois. Dans ce cas, `screen -r` ne sera pas suffisant car on vous demandera de préciser quelle session vous voulez récupérer :

Code : Console

```
$ screen -r
There are several suitable screens on:
        20930.pts-0.mateo21-desktop     (Detached)
        19713.pts-0.mateo21-desktop     (Detached)
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
```

Pour récupérer la session n° 20930, tapez simplement :

Code : Console

```
$ screen -r 20930
```

À noter aussi que `screen -ls` affiche la liste des screens actuellement ouverts :

Code : Console

```
$ screen -ls
There are screens on:
        20930.pts-0.mateo21-desktop     (Detached)
        19713.pts-0.mateo21-desktop     (Detached)
2 Sockets in /var/run/screen/S-mateo21.
```

Certaines personnes ont pris l'habitude de tout faire sur `screen`, notamment sur les serveurs. Il m'est arrivé de laisser tourner une session `screen` pendant plusieurs mois grâce à la possibilité de détachement que nous venons de découvrir.

Un fichier personnalisé de configuration de `screen`

Sans rentrer dans le détail car ce serait bien trop long, sachez qu'il est possible de personnaliser `screen` avec un fichier de configuration, comme la plupart des autres programmes sous Linux d'ailleurs.

Ce fichier s'appelle `.screenrc` et doit être placé dans votre home (`/home/mateo21` par exemple). Vous pouvez vous amuser à lire la doc à ce sujet, mais vous pouvez aussi utiliser le même fichier `.screenrc` que j'ai l'habitude d'utiliser (ce fichier de configuration n'est pas de moi, merci donc à son auteur, « bennyben »).

Une fois placé dans votre home, exécutez `screen`. Vous devriez noter quelques différences, comme vous le montre la figure suivante.

```
mateo21@mateo21-desktop:~$ ls
bin            Examples                  Images      video.avi
cours_unix.txt ies4linux-2.99.0.1        sortiefind  video_copie.avi
Desktop        ies4linux-latest.tar.gz   tuto
mateo21@mateo21-desktop:~$ 















22:13:56 mateo21-desktop 0,50 0,43 0,24 0$ bash  1-$ bash
```

Je trouve cette configuration plus pratique car on a toujours en bas l'heure, le nom de la machine sur laquelle on se trouve, la charge ainsi que la liste des fenêtres ouvertes. Après, libre à vous d'utiliser la configuration par défaut ou celle-là : dans tous les cas, les commandes restent les mêmes. ;-)

En résumé

- Il est possible d'envoyer des programmes en arrière-plan dans la console afin de garder la main pour lancer de nouvelles commandes.
- Pour lancer un processus en arrière-plan, on peut ajouter le symbole `&` à la fin de la commande. En revanche, lorsque vous fermez la console, le processus est arrêté. Si vous voulez qu'il continue, utilisez plutôt la commande `nohup`.
- Si vous avez lancé une commande normalement (en avant-plan) mais que celle-ci s'éternise, vous pouvez utiliser le raccourci `Ctrl + Z` pour la mettre en pause et récupérer la main. Si vous lancez la commande `bg` ensuite, elle reprendra son exécution en arrière-plan. Vous pourrez la récupérer au premier plan avec `fg` à tout moment.
- `screen` est un programme puissant que vous pouvez installer avec `apt-get` (il n'est pas présent par défaut). Il permet d'ouvrir plusieurs consoles virtuelles au sein d'une seule et même console, et donc d'exécuter facilement plusieurs processus en parallèle.

 Exécuter un programme à une heure différée

Noussavons lancer une commande pour qu'elle s'exécute tout de suite. Il est cependant aussi possible de « retarder » son lancement.

Linux vous propose toute une série d'outils qui vous permettent de programmer à l'avance l'exécution d'une tâche, comme par exemple la **crontab** que nous allons étudier.
Tous les outils que nous allons découvrir dans ce chapitre feront en outre appel à la notion de date. Nous allons donc dans un premier temps nous intéresser au formatage de la date.

date : régler l'heure

Nous commencerons par nous intéresser à la date et l'heure du moment, car tout dans ce chapitre tourne autour de la notion de date. ;-)

Je vous ai déjà présenté brièvement la commande `date`. Essayez-la à nouveau :

Code : Console

```
$ date
mercredi 10 novembre 2010, 12:27:25 (UTC+0100)
```

Sans paramètre, la commande nous renvoie donc la date actuelle, l'heure et le décalage horaire.

Personnaliser l'affichage de la date

Si vous regardez le manuel (`man date`), vous verrez qu'il est possible de personnaliser l'affichage de la date : vous pouvez choisir quelles informations vous voulez afficher et dans quel ordre (vous pouvez par exemple ajouter les nanosecondes ou encore le numéro du siècle actuel).

Pour spécifier un affichage personnalisé, vous devez utiliser un symbole + suivi d'une série de symboles qui indiquent l'information que vous désirez. Je vous recommande de mettre le tout entre guillemets.

Prenons quelques exemples pour bien comprendre :

Code : Console

```
$ date "+%H"
12
```

Le `+%H` est le **format de date**. `%H` signifie « le numéro de l'heure actuelle ». Il était donc 12 heures au moment où j'ai lancé la commande.

Essayons autre chose d'un peu plus compliqué :

Code : Console

```
$ date "+%H:%M:%S"
12:36:15
```

Ici, j'ai rajouté les minutes (`%M`) et les secondes (`%S`).
J'ai séparé les nombres par des deux-points, mais j'aurais très bien pu mettre autre chose à la place :

Code : Console

```
$ date "+%Hh%Mm%Ss"
```

```
12h41m01s
```

Seule la lettre qui suit le % est interprétée. Mes lettres « h », « m » et « s » sont donc simplement affichées.


Mais comment tu sais que %M affiche le nombre de minutes, par exemple ?

Je lis le man de date, tout simplement.

C'est là que j'apprends comment afficher l'année, notamment :

Code : Console

```
$ date "+Bienvenue en %Y"
Bienvenue en 2010
```

À vous de jouer !

Modifier la date

La commande date permet aussi de changer la date.


Attention, il faudra être root pour faire cela (vous devrez placer un sudo devant par exemple).

Il faut préciser les informations sous la forme suivante : MMDDhhmmYYYY. Les lettres signifient :

- MM : mois ;
- DD : jour ;
- hh : heure ;
- mm : minutes ;
- YYYY : année.

Notez qu'il n'est pas obligatoire de préciser l'année. On peut donc écrire :

Code : Console

```
$ sudo date 11101250
mercredi 10 novembre 2010, 12:50:00 (UTC+0100)
```

La nouvelle date s'affiche automatiquement et elle est mise à jour sur le système.
Attention à bien respecter l'ordre des nombres : Mois - Jour - Heure - Minutes.

at : exécuter une commande plus tard

Vous souhaitez qu'une commande soit exécutée plus tard ? Pas de problème ! Il est possible de programmer l'exécution d'une commande avec at.


Avec at, le programme ne sera exécuté qu'**une seule fois**. Si vous voulez que l'exécution soit répétée régulièrement, il faudra utiliser la crontab que nous verrons plus loin.

Exécuter une commande à une heure précise

La commande s'utilise en deux temps.

1. Vous indiquez à quel moment (quelle heure, quel jour) vous désirez que la commande soit exécutée.
2. Vous tapez ensuite la commande que vous voulez voir exécutée à l'heure que vous venez d'indiquer.

Il faut donc d'abord indiquer à quelle heure vous voulez exécuter votre commande, sous la forme `HH:MM` :

Code : Console

```
$ at 14:17
```

L'exécution des commandes est demandée à 14 h 17 aujourd'hui. Si vous tapez cela dans votre console, vous devriez voir ceci s'afficher :

Code : Console

```
$ at 14:17
warning: commands will be executed using /bin/sh
at>
```

`at` comprend que vous voulez exécuter des commandes à 14 h 17 et vous demande lesquelles. C'est pour cela qu'un **prompt** est affiché : on vous demande de taper les commandes que vous voulez exécuter à cette heure-là.

Pour cet exemple, nous allons demander de créer un fichier à 14 h 17 :

Code : Console

```
$ at 14:17
warning: commands will be executed using /bin/sh
at> touch fichier.txt
at> <EOT>
job 5 at Mon Nov 10 14:17:00 2010
```

Après avoir écrit la commande `touch`, `at` affiche à nouveau un prompt et vous demande une autre commande. Vous pouvez indiquer une autre commande à exécuter à la même heure… ou bien arrêter là. Dans ce cas, tapez `Ctrl + D` (comme si vous cherchiez à sortir d'un terminal). Le symbole `<EOT>` devrait alors s'afficher, et `at` s'arrêtera.

`at` affiche ensuite le numéro associé à cette tâche (à ce « job », comme il dit) et l'heure à laquelle il sera exécuté.

Attendez 14 h 17, et vous verrez que le fichier sera créé. :-)



Et si je veux exécuter la commande demain à 14 h 17 et non pas aujourd'hui ?

Code : Console

```
$ at 14:17 tomorrow
```

`tomorrow` signifie « demain ».



Et si je veux exécuter la commande le 15 novembre à 14 h 17 ?

Code : Console

```
$ at 14:17 11/15/10
```



La date est au format américain, les numéros du jour et du mois sont donc inversés : 11/15/10. 11 correspond au mois (novembre) et 15 au numéro du jour !

Exécuter une commande après un certain délai

Il est possible d'exécuter une commande dans 5 minutes, 2 heures ou 3 jours sans avoir à écrire la date.
Par exemple, pour exécuter la commande dans 5 minutes :

Code : Console

```
$ at now +5 minutes
```

… ce qui signifie « Dans maintenant (*now*) + 5 minutes ». Les mots-clés utilisables sont les suivants :

- `minutes` ;
- `hours` (heures) ;
- `days` (jours) ;
- `weeks` (semaines) ;
- `months` (mois) ;
- `years` (années).

Un autre exemple :

Code : Console

```
$ at now +2 weeks
```

… exécutera les commandes dans deux semaines.

`atq` et `atrm` : lister et supprimer les *jobs* en attente

Chaque fois qu'une commande est « enregistrée », `at` nous indique un numéro de job ainsi que l'heure à laquelle il sera exécuté.

Il est possible d'avoir la liste des jobs en attente avec la commande `atq` :

Code : Console

```
$ atq
13        Mon Nov 10 14:44:00 2010 a mateo21
12        Mon Nov 10 14:42:00 2010 a mateo21
```

Si vous souhaitez supprimer le job n° 13 (je ne sais pas, parce que ça porte malheur par exemple), utilisez `atrm` :

Code : Console

```
$ atrm 13
```

sleep : faire une pause

Le saviez-vous ? Vous pouvez enchaîner plusieurs commandes à la suite en les séparant par des points-virgules comme ceci :

Code : Console

```
$ touch fichier.txt; rm fichier.txt
```

`touch` est d'abord exécuté, puis une fois qu'il a fini ce sera le tour de `rm` (qui supprimera le fichier que nous venons de créer).

Parfois, enchaîner les commandes comme ceci est bien pratique… mais on a besoin de faire une pause entre les deux.
C'est là qu'intervient `sleep` : cette commande permet de faire une pause.

Code : Console

```
$ touch fichier.txt; sleep 10; rm fichier.txt
```

Cette fois, il va se passer les choses suivantes :

- `fichier.txt` est créé ;
- `sleep` fait une pause de 10 secondes ;
- `rm` supprime ensuite le fichier.

Par défaut, la pause est exprimée en secondes. Il est aussi possible d'utiliser d'autres symboles pour changer l'unité :

- `m` : minutes ;
- `h` : heures ;
- `d` : jours.

Pour faire une pause d'une minute :

Code : Console

```
$ touch fichier.txt; sleep 1m; rm fichier.txt
```

L'intérêt de `sleep` ne vous paraîtra peut-être pas évident tout de suite, mais retenez que cette commande existe car il est parfois bien pratique de faire une pause, par exemple pour s'assurer que la première commande a bien eu le temps de se terminer. ;-)



Vous pouvez aussi remplacer les points-virgules par des `&&`, comme ceci :

Code : Console

```
touch fichier.txt && sleep 10 && rm fichier.txt
```

Dans ce cas, les instructions ne s'enchaîneront que si elles se sont correctement exécutées. Par exemple, si `touch` renvoie une erreur pour une raison ou une autre, alors les commandes qui suivent (`sleep`, `rm`) ne seront pas exécutées.

crontab : exécuter une commande régulièrement

La « crontab » constitue un incontournable sous Linux : cet outil nous permet de programmer l'exécution régulière d'un programme.
Contrairement à `at` qui n'exécutera le programme qu'une seule fois, `crontab` permet de faire en sorte que l'exécution soit répétée : toutes les heures, toutes les minutes, tous les jours, tous les trois jours, etc.

Un peu de configuration…

Avant toute chose, nous devons modifier notre configuration (notre fichier `.bashrc`) pour demander à ce que Nano soit l'éditeur par défaut. En général, c'est le programme « vi » qui fait office d'éditeur par défaut. C'est un bon éditeur de texte, mais bien plus complexe que Nano et je ne vous le présenterai que plus tard.
En attendant, rajoutez la ligne suivante à la fin de votre fichier `.bashrc` :

Code : Console

```
export EDITOR=nano
```

Vous pouvez aussi écrire la commande suivante :

Code : Console

```
$ echo "export EDITOR=nano" >>  ~/.bashrc
```

Cela aura pour effet d'écrire cette ligne à la fin de votre fichier `.bashrc` situé dans votre répertoire personnel.
Fermez ensuite votre console et rouvrez-la pour que cette nouvelle configuration soit bien prise en compte.

Cette petite configuration étant faite, attaquons les choses sérieuses.

La « crontab », qu'est-ce que c'est ?

`crontab` est en fait une commande qui permet de lire et de modifier un fichier appelé la « crontab ».
Ce fichier contient la liste des programmes que vous souhaitez exécuter régulièrement, et à quelle heure vous souhaitez qu'ils soient exécutés.



`crontab` permet donc de changer la liste des programmes régulièrement exécutés. C'est toutefois le programme `cron` qui se charge d'exécuter ces programmes aux heures demandées.
Ne confondez donc pas `crontab` et `cron` : le premier permet de modifier la liste des programmes à exécuter, le second les exécute.

Comment utilise-t-on `crontab` ?
Il y a trois paramètres différents à connaître, pas plus :

- `-e` : modifier la crontab ;
- `-l` : afficher la crontab actuelle ;
- `-r` : supprimer votre crontab. Attention, la suppression est immédiate et sans confirmation !

Commençons par afficher la crontab actuelle :

Code : Console

```
$ crontab -l
no crontab for mateo21
```

Normalement, vous n'avez pas encore créé de crontab. Vous noterez qu'il y a une crontab par utilisateur. Là j'édite la crontab de mateo21 car je suis loggé avec l'utilisateur mateo21, mais root a aussi sa propre crontab. La preuve :

Code : Console

```
$ sudo crontab -l
no crontab for root
```

Bien, intéressons-nous à la modification de la crontab. Tapez :

Code : Console

```
$ crontab -e
```

Si vous avez bien configuré votre `.bashrc` tout à l'heure (et que vous avez relancé votre console), cela devrait ouvrir le programme Nano que vous connaissez.
Si par hasard vous n'avez pas fait quelque chose correctement, c'est le programme « vi » qui se lancera. Comme vous ne le connaissez pas encore, tapez `:q` puis `Entrée` pour sortir. Vérifiez à nouveau votre configuration du `.bashrc` et n'oubliez pas de fermer puis de rouvrir votre console.

Modifier la crontab

Pour le moment, si votre crontab est vide comme la mienne, vous devriez voir uniquement ceci (capture d'écran de Nano) :

Code : Console

```
GNU nano 2.0.7        Fichier : /tmp/crontab.4u4jHU/crontab

# m h  dom mon dow   command

                             [ Lecture de 1 ligne ]
^G Aide       ^O Écrire    ^R Lire fich.^Y Page préc.^K Couper    ^C Pos. cur.
^X Quitter    ^J Justifier ^W Chercher  ^V Page suiv.^U Coller    ^T Orthograp.
```

Les champs

Le fichier ne contient qu'une seule ligne :

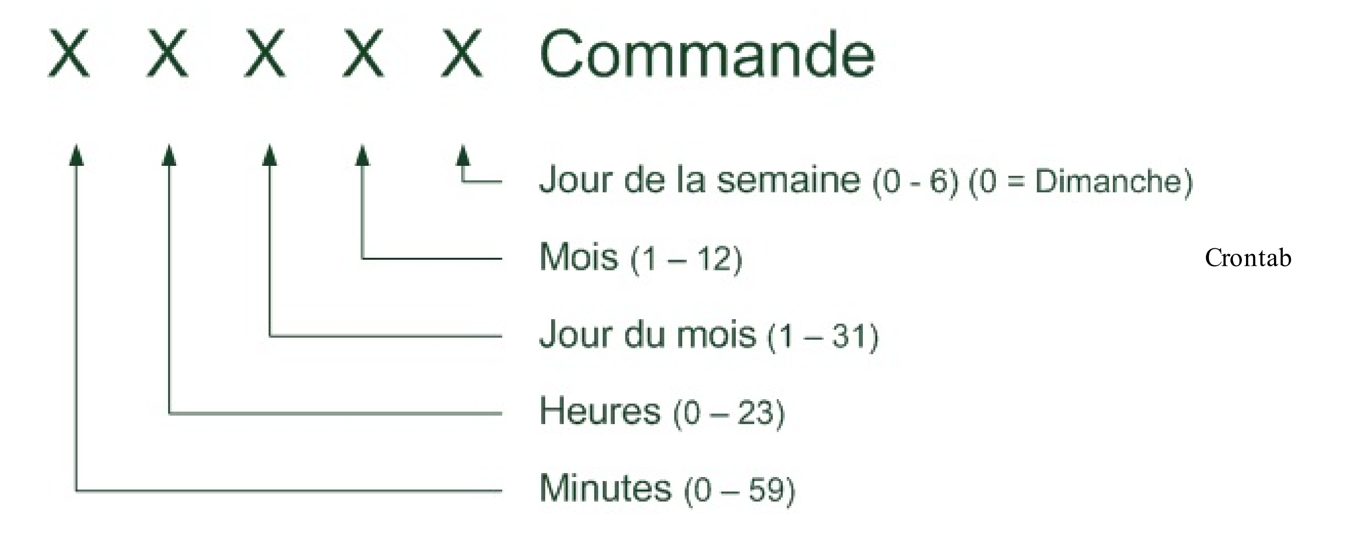
Code : Console

```
# m h  dom mon dow   command
```

Comme cette ligne est précédée d'un `#`, il s'agit d'un commentaire (qui sera donc ignoré).
Cette ligne vous donne quelques indications sur la syntaxe du fichier :

- `m` : minutes (0 - 59) ;
- `h` : heures (0 - 23) ;
- `dom` (*day of month*) : jour du mois (1 - 31) ;
- `mon` (*month*) : mois (1 - 12) ;
- `dow` (*day of week*) : jour de la semaine (0 - 6, 0 étant le dimanche) ;
- `command` : c'est la commande à exécuter.

Chaque ligne du fichier correspond à une commande que l'on veut voir exécutée régulièrement. Vous trouverez en figure suivante un schéma qui résume la syntaxe d'une ligne.



Crontab

En clair, vous devez d'abord indiquer à quel moment vous voulez que la commande soit exécutée, puis ensuite écrire à la fin la commande à exécuter.
C'est un peu comme un tableau. Chaque champ est séparé par un espace.

Chaque « X » sur mon schéma peut être remplacé soit par un nombre, soit par une étoile qui signifie « tous les nombres sont valables ».

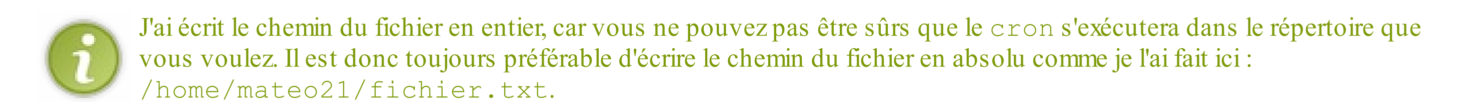
Bien comprendre la crontab n'est pas si simple, je vous propose donc de nous baser sur quelques exemples pour voir comment ça fonctionne.

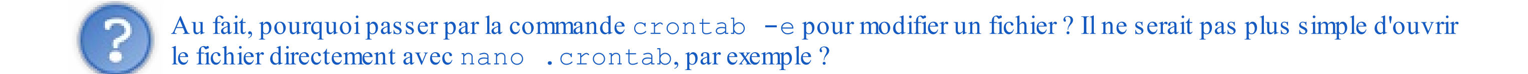
Imaginons que je veuille exécuter une commande tous les jours à 15 h 47. Je vais écrire ceci :

Code : Console

```
47 15 * * * touch /home/mateo21/fichier.txt
```

Seules les deux premières valeurs sont précisées : les minutes et les heures. Chaque fois qu'il est 15 h 47, la commande indiquée à la fin sera exécutée.


J'ai écrit le chemin du fichier en entier, car vous ne pouvez pas être sûrs que le `cron` s'exécutera dans le répertoire que vous voulez. Il est donc toujours préférable d'écrire le chemin du fichier en absolu comme je l'ai fait ici : `/home/mateo21/fichier.txt`.


Au fait, pourquoi passer par la commande `crontab -e` pour modifier un fichier ? Il ne serait pas plus simple d'ouvrir le fichier directement avec `nano .crontab`, par exemple ?

Oui, mais ce n'est pas comme cela que ça fonctionne. La crontab exige de passer par une commande, c'est comme ça.
Il y a quelques avantages à cela, puisque cela permet au programme de vérifier si votre fichier est correctement écrit avant de mettre à jour la crontab. S'il y a une erreur de syntaxe, on vous le dira et aucun changement ne sera apporté.

Essayez d'enregistrer et de quitter Nano. Vous verrez que la crontab vous dit qu'elle « installe » les changements (elle les prend en compte, en quelque sorte) :

Code : Console

```
crontab: installing new crontab
mateo21@mateo21-desktop:~$
```

Désormais, `fichier.txt` sera créé dans mon répertoire personnel tous les jours à 15 h 47 (s'il n'existe pas déjà).

Revenez dans la crontab, nous allons voir d'autres exemples (tableau suivante).

Crontab	Signification
`47 * * * * commande`	Toutes les heures à 47 minutes exactement.> & Donc à 00 h 47, 01 h 47, 02 h 47, etc.
`0 0 * * 1 commande`	Tous les lundis à minuit (dans la nuit de dimanche à lundi).
`0 4 1 * * commande`	Tous les premiers du mois à 4 h du matin.
`0 4 * 12 * commande`	Tous les jours du mois de décembre à 4 h du matin.
`0 * 4 12 * commande`	Toutes les heures les 4 décembre.
`* * * * * commande`	Toutes les minutes !



Est-il possible d'exécuter une commande plus fréquemment que toutes les minutes ?

Non, c'est impossible avec `cron`. La fréquence minimale, c'est toutes les minutes.

Les différentes notations possibles

Pour chaque champ, on a le droit à différentes notations :

- `5` (un nombre) : exécuté lorsque le champ prend la valeur 5 ;
- `*` : exécuté tout le temps (toutes les valeurs sont bonnes) ;
- `3,5,10` : exécuté lorsque le champ prend la valeur 3, 5 ou 10. Ne pas mettre d'espace après la virgule ;
- `3-7` : exécuté pour les valeurs 3 à 7 ;
- `*/3` : exécuté tous les multiples de 3 (par exemple à 0 h, 3 h, 6 h, 9 h…).

Vous connaissiez déjà les deux premières notations. Celles que nous venons de découvrir nous permettent de démultiplier les possibilités offertes par la crontab.
Voici, sur le tableau suivante, quelques exemples d'utilisation.

Crontab	Signification
`30 5 1-15 * * commande`	À 5 h 30 du matin du 1er au 15 de chaque mois.
`0 0 * * 1,3,4 commande`	À minuit le lundi, le mercredi et le jeudi.
`0 */2 * * * commande`	Toutes les 2 heures (00 h 00, 02 h 00, 04 h 00…)
`*/10 * * * 1-5 commande`	Toutes les 10 minutes du lundi au vendredi.

Comme vous le voyez, la crontab offre de très larges possibilités (pour peu que l'on ait compris comment elle fonctionne).

Rediriger la sortie

Pour le moment, nous avons exécuté notre commande très simplement dans la crontab :

Code : Console

```
47 15 * * * touch /home/mateo21/fichier.txt
```

Toutefois, il faut savoir que si la commande renvoie une information ou une erreur, vous ne la verrez pas apparaître dans la

console. Normal : ce n'est pas vous qui exécutez la commande, mais le programme `cron`.

Que se passe-t-il alors si la commande renvoie un message ? En fait, le résultat de la commande vous est envoyé par e-mail. Chaque utilisateur possède sa propre boîte e-mail sur les machines de type Unix, mais je ne vais pas m'attarder là-dessus. Nous allons plutôt voir comment rediriger le résultat.

Tenez : rediriger une sortie, vous savez faire ça, non ?

Code : Console

```
47 15 * * * touch /home/mateo21/fichier.txt >> /home/mateo21/cron.log
```

Tous les messages seront désormais ajoutés à la fin de `cron.log`. Tous ? Non, on oublie d'y rediriger aussi les erreurs !

Code : Console

```
47 15 * * * touch /home/mateo21/fichier.txt >> /home/mateo21/cron.log 2>&1
```

Voilà, c'est mieux.
Cette fois, tout sera envoyé dans `cron.log` : les messages et les erreurs.



Et si je ne veux pas du tout récupérer ce qui est affiché ?

Nous avons déjà appris à le faire ! Il suffit de rediriger dans `/dev/null` (le fameux « trou noir » du système). Tout ce qui est envoyé là-dedans est immédiatement supprimé : hop, plus de trace, le crime parfait.

Code : Console

```
47 15 * * * touch /home/mateo21/fichier.txt > /dev/null 2>&1
```

En résumé

- `date` permet d'obtenir la date et l'heure mais aussi de modifier celles-ci.
- `at` retarde l'exécution d'une commande à une heure ultérieure.
- On peut exécuter plusieurs commandes d'affilée en les séparant par des points-virgules : `touch fichier.txt; rm fichier.txt`.
- La commande `sleep` permet de faire une pause entre deux commandes exécutées d'affilée.
- `crontab` permet de programmer des commandes pour une exécution régulière. Par exemple : tous les jours à 18 h 30, tous les lundis et mardis à 12 h, tous les 5 du mois, etc. On modifie la programmation avec `crontab -e`.

Partie 4 : Transférer des données à travers le réseau

 Archiver et compresser

Pour bien débuter cette partie sur le réseau, il me semble logique de vous présenter d'abord le fonctionnement de la compression sous Linux. En effet, si vous vous apprêtez à envoyer un ou plusieurs fichiers par le réseau (que ce soit par mail, FTP ou autre), il est toujours préférable de commencer par les compresser afin de réduire leur taille.

Vous avez sûrement déjà entendu parler du format `zip`. C'est le plus connu et le plus répandu… du moins sous Windows. On peut l'utiliser aussi sous Linux, de même que le format `rar`.

Cependant, on préfèrera utiliser des alternatives libres (et souvent plus puissantes) telles que le `gzip` et le `bzip2`. Toutefois, contrairement à `zip` et `rar`, le `gzip` et le `bzip2` ne sont capables de compresser qu'un seul fichier à la fois et ne peuvent donc pas créer un « paquetage » de plusieurs fichiers.
Mais rassurez-vous, tout est prévu : on utilise pour cela un outil à part, appelé `tar`, qui permet d'assembler des fichiers avant de les compresser.

Nous allons découvrir le fonctionnement de tout cela dans ce chapitre. ;-)

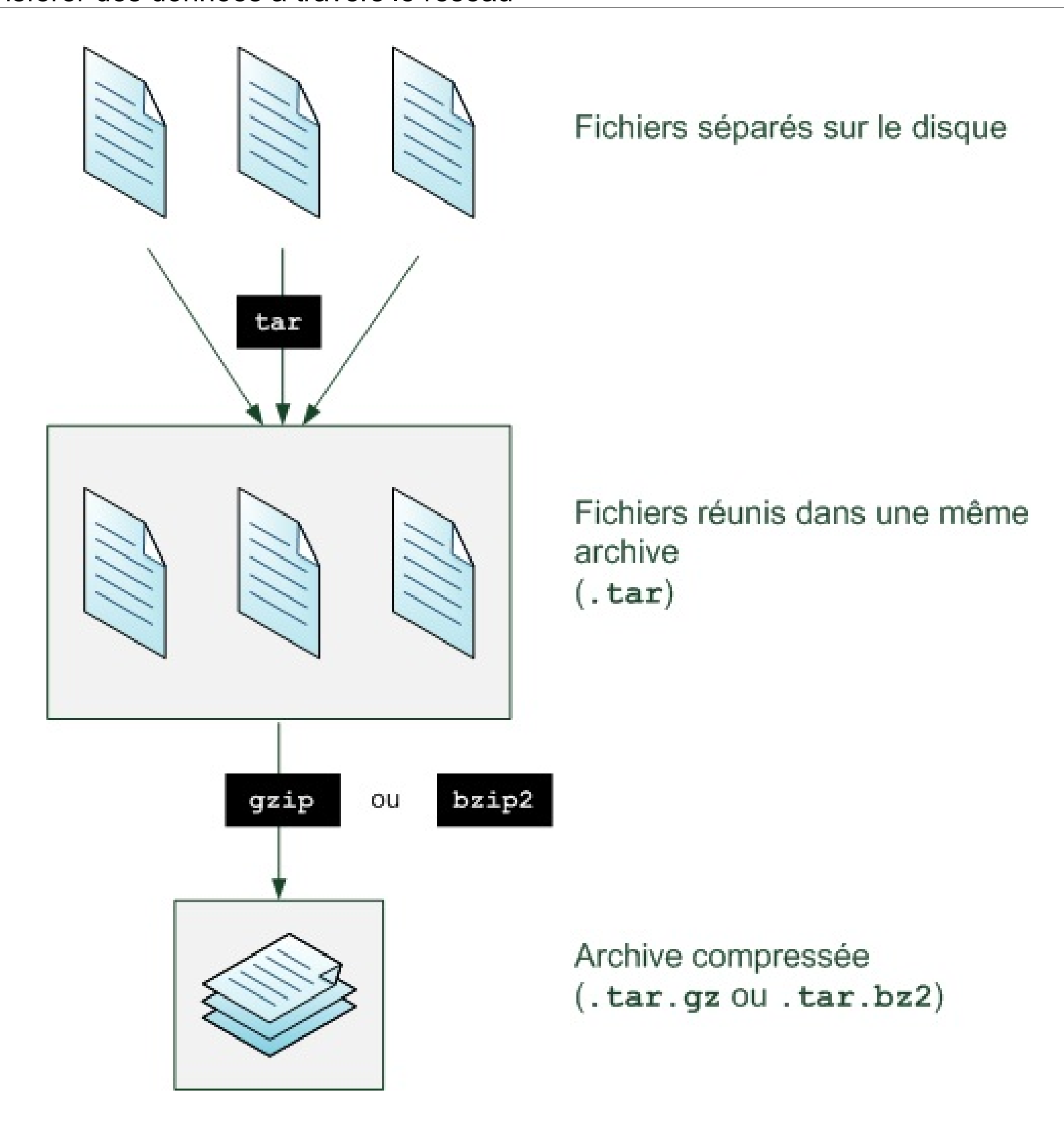
tar : assembler des fichiers dans une archive

Comme je vous le disais en introduction, aussi étonnant que cela puisse paraître, les `gzip` et `bzip2` ne permettent de compresser qu'un seul fichier à la fois. Comment faire alors si vous voulez compresser une dizaine de fichiers ?

Sous Linux, on a depuis longtemps pris l'habitude de procéder en deux étapes :

1. réunir les fichiers dans un seul gros fichier appelé **archive**. On utilise pour cela le programme `tar` ;
2. compresser le gros fichier ainsi obtenu à l'aide de `gzip` ou de `bzip2`.

Ces deux étapes sont résumées dans le schéma de la figure suivante.



Nous allons dans un premier temps apprendre à manipuler `tar`, puis nous verrons la compression avec `gzip` et `bzip2`, sans oublier les formats `zip` et `rar` que vos amis utilisant Windows risquent de vous envoyer un jour ou l'autre.



Les formats `zip` et `rar` ne séparent pas les étapes comme sur le schéma suivant. Ils sont capables d'assembler plusieurs fichiers en une archive et de la compresser en même temps.

Comme vous le voyez, sous Linux il y a donc une méthode à suivre dans un ordre précis. Voyons ensemble comment faire !

Regrouper d'abord les fichiers dans un même dossier

Vous avez plusieurs fichiers que vous souhaitez compresser. Dans mon cas, ce sont des fichiers `.tuto` (qui contiennent des chapitres de tutoriels du Site du Zéro), mais vous pouvez bien entendu compresser ce que vous voulez : des textes, présentations, tableurs, *logs*, etc.



Certains formats de fichier sont déjà compressés. C'est le cas des images jpeg, png et gif, mais aussi de la plupart des vidéos. Vous pouvez bien entendu les assembler dans une archive `tar` et même les compresser, mais vous ne les rendrez en général pas plus petits car ils ont déjà été compressés.

Mes fichiers `.tuto` que je souhaite archiver sont pour le moment placés en vrac dans mon home :

Code : Console

```
$ ls
Bureau      Images                              1-heritage.tuto   Public
```

```
Documents   la-surcharge-d-operateurs.tuto  Modèles          Vidéos
Examples    les-principaux-widgets.tuto     Musique
```

Il est recommandé de placer d'abord les fichiers à archiver dans un seul et même dossier. Créons-le et déplaçons-y tous nos .tuto :

Code : Console

```
$ mkdir tutoriels
$ mv *.tuto tutoriels/
$ ls
Bureau      Examples  Modèles  Public     Vidéos
Documents   Images    Musique  tutoriels
```

Voilà, nos fichiers sont réunis dans le dossier tutoriels.

-cvf : créer une archive tar

Nous allons maintenant créer une archive tar de ce dossier et de ses fichiers. La procédure à suivre pour créer une archive est :

Code : Console

```
tar -cvf nom_archive.tar nom_dossier/
```

J'utilise trois options :

- -c : signifie créer une archive tar ;
- -v : signifie afficher le détail des opérations ;
- -f : signifie assembler l'archive dans un fichier.

Essayons de faire cela sur notre dossier tutoriels :

Code : Console

```
$ tar -cvf tutoriels.tar tutoriels/
tutoriels/
tutoriels/les-principaux-widgets.tuto
tutoriels/la-surcharge-d-operateurs.tuto
tutoriels/l-heritage.tuto
```

Ici on archive le dossier tutoriels et donc son contenu. Grâce à -v, on voit bien la liste des fichiers qui ont été archivés.



Est-on obligé de mettre systématiquement nos fichiers dans un même dossier pour archiver ensuite ce dossier ? On ne pourrait pas archiver directement les fichiers ?

Si, c'est possible. Imaginons que nous soyons toujours dans notre home avec nos fichiers .tuto. On pourrait très bien faire :

Code : Console

```
tar -cvf archive.tar fichier1 fichier2 fichier3
```

C'est possible et ça fonctionne. Toutefois, il est de coutume sous Linux de placer **d'abord** les fichiers dans un dossier avant de les mettre dans le `tar`. Cela permet d'éviter, lorsqu'on extrait les fichiers de l'archive, que ceux-ci aillent se mêler à d'autres fichiers. Nous allons voir ce problème maintenant.

`-tf` : afficher le contenu de l'archive sans l'extraire

Vous venez de recevoir une archive `tar` qu'on vous a envoyée. Bien. Mais que contient-elle ?
Avant d'extraire quoi que ce soit, vous aimeriez peut-être voir son contenu. C'est possible avec `-tf` :

Code : Console

```
$ tar -tf tutoriels.tar
tutoriels/
tutoriels/les-principaux-widgets.tuto
tutoriels/la-surcharge-d-operateurs.tuto
tutoriels/l-heritage.tuto
```

Quand on fait cela, on voit que tous les fichiers sont réunis dans un même dossier `tutoriels`, et ça c'est très pratique.
J'en reviens justement au problème dont je parlais un peu plus haut : imaginez que vous « détariez » une archive contenant plus de 400 fichiers dans votre home. Si ces fichiers n'étaient pas réunis dans un dossier, ils iraient tous se mêler aux autres fichiers du home, et alors là, je vous dis pas la pagaille !

Voilà donc pourquoi je vous ai invités dès le début à réunir vos fichiers à archiver dans un même dossier. Cela permet d'éviter de mauvaises surprises à celui qui extrait les fichiers de l'archive. Quasiment toutes les archives que l'on vous proposera de télécharger suivent ce même schéma et font attention à tout réunir dans un même dossier, mais vérifiez le contenu avant de l'extraire, on ne sait jamais !

`-rvf` : ajouter un fichier

Si vous avez oublié un fichier, vous pouvez toujours l'ajouter par la suite avec `-rvf` :

Code : Console

```
$ tar -rvf tutoriels.tar fichier_supplementaire.tuto
tutoriels/fichier_supplementaire.tuto
```

`-xvf` : extraire les fichiers de l'archive

Pour extraire les fichiers, on va utiliser les options `-xvf` (`-x` pour *eXtract*) :

Code : Console

```
$ tar -xvf tutoriels.tar
tutoriels/
tutoriels/les-principaux-widgets.tuto
tutoriels/la-surcharge-d-operateurs.tuto
tutoriels/l-heritage.tuto
```

Les fichiers s'extraient dans le répertoire dans lequel vous vous trouvez. Vérifiez donc avant de les extraire que ceux-ci sont réunis dans un même dossier (avec `-tf`) si vous ne voulez pas que ces fichiers aillent se mélanger à d'autres !

gzip & bzip2 : compresser une archive

Vous avez maintenant créé une belle archive `tar`. Tous vos fichiers sont réunis là-dedans. Voyons comment compresser cela.

Nous disposons de deux programmes de compression bien répandus dans le monde Linux :

- `gzip` : c'est le plus connu et le plus utilisé ;
- `bzip2` : il est un peu moins fréquemment utilisé. Il compresse mieux mais plus lentement que `gzip`.



À noter qu'il existe aussi le vieux programme `compress`. Il n'est cependant plus vraiment utilisé car on dispose aujourd'hui de meilleurs algorithmes de compression : `gzip` et `bzip2`.

Ces programmes sont simples à utiliser. Ils prennent comme paramètre le nom du fichier à compresser. Ils le compressent et modifient ensuite son nom.
Concrètement, ils ajoutent un suffixe pour indiquer que l'archive a été compressée :

- `.tar.gz` : si l'archive a été compressée avec `gzip` ;
- `.tar.bz2` : si l'archive a été compressée avec `bzip2`.

À titre indicatif, voici les différentes tailles de l'archive, avant et après compression :

Fichier	Taille
`tutoriels.tar`	130 Ko
`tutoriels.tar.gz`	35 Ko
`tutoriels.tar.bz2`	29 Ko

Cela confirme ce que je vous disais : `bzip2` est plus efficace… mais il compresse aussi plus lentement et est moins fréquemment utilisé.

gzip : la compression la plus courante

Concrètement, le programme `gzip` s'utilise de la manière la plus simple qui soit :

Code : Console

```
gzip tutoriels.tar
```

L'archive est compressée et gagne ensuite le suffixe `.gz`. Elle s'appelle donc désormais `tutoriels.tar.gz`. Voilà pourquoi vous voyez circuler sur l'internet des fichiers `.tar.gz` : cela signifie que ce sont des archives compressées !

Pour décompresser l'archive ensuite, il suffit d'utiliser `gunzip` :

Code : Console

```
gunzip tutoriels.tar.gz
```

L'archive retrouve son état non compressé en `.tar`. Vous pouvez maintenant extraire les fichiers de l'archive comme vous avez appris à le faire un peu plus tôt avec `tar -xvf`. ;-)

bzip2 : la compression la plus puissante

Le fonctionnement de `bzip2` est le même que celui de `gzip` :

Code : Console

```
bzip2 tutoriels.tar
```

Une archive compressée `tutoriels.tar.bz2` sera alors créée. Pour la décompresser, utilisez `bunzip2` :

Code : Console

```
bunzip2 tutoriels.tar.bz2
```

Vous retrouvez un `.tar` que vous pouvez extraire avec `tar -xvf`.

Archiver et compresser en même temps avec `tar`



C'est bien beau de séparer les étapes, mais cela nous demande de taper deux fois plus de commandes pour compresser et décompresser des fichiers ! Il n'y a pas plus rapide ?

Si on fait comme cela, c'est essentiellement pour des raisons historiques. Souvenez-vous que Linux ne fait que recopier le fonctionnement d'Unix dont l'origine remonte aux années 1960 !

Heureusement, les choses ont un peu évolué. Il faut toujours archiver puis compresser, mais le programme `tar` est capable d'appeler lui-même `gzip` ou `bzip2` si vous lui donnez les bons paramètres.

`-zcvf` : archiver et compresser en `gzip`

Vous connaissez `tar -cvf` pour créer une archive `tar`. Si vous rajoutez l'option `-z`, l'archive sera automatiquement compressée avec `gzip`.

Code : Console

```
tar -zcvf tutoriels.tar.gz tutoriels/
```

Voilà comment on obtient une archive compressée en une seule commande. :-)

Pour décompresser, c'est pareil, sauf que le `-c` est remplacé par un `-x` comme tout à l'heure :

Code : Console

```
tar -zxvf tutoriels.tar.gz
```

`-jcvf` : archiver et compresser en `bzip2`

Le principe est le même avec `-j` à la place de `-z` :

Code : Console

```
tar -jcvf tutoriels.tar.bz2 tutoriels/
```

Et pour extraire :

Code : Console

```
tar -jxvf tutoriels.tar.bz2 tutoriels/
```



Vous pouvez toujours analyser le contenu de l'archive avant de la décompresser. Avec `-ztf`, vous regarderez à l'intérieur d'une archive « gzippée » et avec `-jtf`, vous regarderez à l'intérieur d'une archive « bzippée-deux ».

`zcat`, `zmore` & `zless` : afficher directement un fichier compressé

Parfois, on compresse non pas une archive `tar` mais directement un fichier. Par exemple, je peux compresser un fichier `.tuto` directement :

Code : Console

```
gzip l-heritage.tuto
```

Le fichier est alors compressé et renommé en `l-heritage.tuto.gz`.
Maintenant, supposons que nous voulions afficher le contenu de ce fichier sans le décompresser auparavant. Eh bien il existe des outils qui permettent de faire cela !

- `zcat` : équivalent de `cat`, capable de lire un fichier compressé (gzippé).
- `zmore` : équivalent de `more`, capable de lire un fichier compressé (gzippé).
- `zless` : équivalent de `less`, capable de lire un fichier compressé (gzippé).

Si vous essayez de faire un `cat l-heritage.tuto.gz`, vous allez voir des caractères bizarres s'afficher à l'écran, comme le montre la figure suivante.

Fichier Édition Affichage Terminal Onglets Aide

Ces caractères bizarres constituent une représentation de votre fichier compressé. Comme vous pouvez le voir, ce n'est pas très lisible.
À ce stade, votre console est d'ailleurs boguée. Si vous tapez des caractères, vous allez voir que vous allez taper n'importe quoi. Pour réinitialiser la console, tapez la commande `reset` puis appuyez sur `Entrée`.

Maintenant, essayez plutôt d'utiliser `zcat`. Ce programme va décompresser le fichier à la volée et l'afficher dans la console :

Code : Console

```
$ zcat l-heritage.tuto.gz
 <conclusion>
    <![CDATA[Ce chapitre en impose peut-
être un peu par sa taille, mais ne vous y fiez pas ce sont surtout les schémas qui
```

`zmore` et `zless`, équivalents de `more` et `less` qui permettent d'afficher page par page, fonctionnent aussi !

unzip & unrar : décompresser les .zip et .rar

Les `.tar.gz` et `.tar.bz2` ont beau être courants dans le monde Linux, vos amis utilisant Windows ne les connaissent pas et risquent tôt ou tard de vous envoyer un superbe `.zip` ou `.rar`… que vous ne pouvez pas décompresser avec `gunzip`.

Heureusement, il existe des utilitaires de décompression pour ces formats. Ils ne sont pas toujours installés par défaut, il faudra donc les installer si vous ne les avez pas.

unzip : décompresser un .zip

Vous venez de recevoir un `.zip` ?
Pas de panique ! Le programme `unzip` est capable de l'extraire. Il est peut-être installé par défaut, mais si vous ne l'avez pas, vous savez ce qu'il vous reste à faire :

Code : Console

```
sudo apt-get install unzip
```

Ceci étant fait, l'utilisation d'`unzip` est très simple :

Code : Console

```
unzip archive.zip
```



Les fichiers vont s'extraire dans le dossier dans lequel vous vous trouvez ! Le problème est le même qu'avec les `.tar.gz` et `.tar.bz2`. Avant de décompresser, vérifiez si les fichiers sont réunis dans un même dossier.

Pour voir le contenu d'une archive `zip` sans l'extraire, utilisez `-l` :

Code : Console

```
$ unzip -l tutoriels.zip
Archive:  tutoriels.zip
  Length     Date   Time    Name
 --------    ----   ----    ----
        0  11-12-08 15:04   tutoriels/
    59515  11-12-08 14:44   tutoriels/les-principaux-widgets.tuto
    36757  11-12-08 14:43   tutoriels/la-surcharge-d-operateurs.tuto
    27685  11-12-08 14:44   tutoriels/l-heritage.tuto
 --------                   -------
   123957                   4 files
```

On peut voir que les fichiers sont réunis dans un même dossier dans l'archive. C'est plutôt rare avec les `.zip` en principe, faites donc attention avant de décompresser les fichiers pour qu'ils n'atterrissent pas n'importe où.

En général, on a surtout besoin d'`unzip` pour décompresser un `zip`, mais il est peu fréquent que l'on soit amené à créer un fichier `zip` (on préfèrera toujours le `gzip` ou le `bzip2`). Si toutefois vous voulez vraiment créer un `zip`, installez le programme `zip` puis basez-vous sur la commande suivante :

Code : Console

```
zip -r tutoriels.zip tutoriels/
```

Le `-r` demande à compresser tous les fichiers contenus dans le dossier tutoriels (sans ce paramètre, seul le dossier, vide, sera compressé !).

`unrar` : décompresser un `.rar`

Il vous faut installer le paquet `unrar` pour pouvoir utiliser cette commande :

Code : Console

```
sudo apt-get install unrar
```

Ensuite, pour extraire :

Code : Console

```
unrar e tutoriels.rar
```

Non, vous ne rêvez pas, l'auteur du programme ne veut pas que l'on mette un tiret devant l'option `e` ! Il faut bien qu'il y ait des exceptions dans la vie. :-)

Pour lister le contenu avant décompression, utilisez l'option `l` :

Code : Console

```
$ unrar l tutoriels.rar

UNRAR 3.80 beta 2 freeware      Copyright (c) 1993-2008 Alexander Roshal

Archive tutoriels.rar

 Name                Size    Packed Ratio  Date    Time     Attr       CRC   Meth Ve
------------------------------------------------------------------------------
 les-principaux-widgets.tuto    59515    16191   27% 12-11-08 14:44 -rw-r--r--
  6E266812 m3b 2.9
 la-surcharge-d-operateurs.tuto    36757    11215   30% 12-11-08 14:43  -rw-r--
r-- E8474528 m3b 2.9
 l-heritage.tuto    27685     8720   31% 12-11-08 14:44 -rw-r--r--
 738EF121 m3b  2.9
------------------------------------------------------------------------------
    3           123957    36126   29%
```



Et si je veux créer des `.rar` ?

Ce n'est pas possible. En fait, le format `rar` est propriétaire. La méthode de décompression a été publiée et vous pouvez donc décompresser des `.rar`, mais pour créer des `.rar` il faut… acheter le logiciel.

Vous pouvez toujours installer le paquet `rar` mais vous verrez que c'est un shareware, qu'il n'est pas libre et qu'il faudra l'acheter sous 40 jours… bref, ce n'est pas le meilleur plan. Si vraiment vous voulez rester compatibles, créez plutôt des `.zip`.

En résumé

- Pour regrouper plusieurs fichiers et dossiers au sein d'un même fichier (appelé *archive*), on utilise le programme `tar`. Celui-ci ne compresse pas les fichiers par défaut, contrairement à `zip`.
- Il est possible de compresser une archive `tar` avec le programme `gzip` (très couramment utilisé) ou `bzip2` (meilleure compression mais plus lente).
- Les archives non compressées ont l'extension `.tar`, les archives compressées ont l'extension `.tar.gz` (pour `gzip`) ou `.tar.bz2` (pour `bzip2`).
- On utilise peu les formats de compression `zip` et `rar` sous Linux, mais il est possible de décompresser ces types de fichiers avec les programmes `unzip` et `unrar`. Ceux-ci ne sont en général pas installés par défaut.

 La connexion sécurisée à distance avec SSH

Voici probablement l'un des chapitres les plus intéressants de ce livre. Nous allons découvrir comment se connecter à distance à une machine équipée de Linux.

Je vous en ai déjà un peu parlé au début de cet ouvrage : toutes les machines sous Linux peuvent être configurées pour que l'on s'y connecte à distance, pour peu qu'elles restent allumées.

Ici, nous n'allons pas seulement découvrir la connexion à distance. Nous allons aussi essayer de comprendre comment cela fonctionne et comment les données sont sécurisées grâce au **protocole SSH**. Ce chapitre sera donc l'occasion de découvrir de nouvelles notions sur le monde passionnant des réseaux et de la sécurité (cryptographie).

Prenons un cas concret : votre ordinateur chez vous est sous Linux, vous le laissez allumé. Pendant la journée au boulot, vous avez besoin de lancer un téléchargement ou de récupérer un document. Vous vous connectez à distance sur votre machine et vous ouvrez une console comme si vous étiez en face de votre PC ! Tout ce que vous avez appris à faire dans une console, vous pouvez le faire à distance depuis n'importe quelle machine dans le monde.



Ce chapitre intéressera en particulier ceux qui ont besoin d'apprendre à gérer un serveur dédié. Ils sont de plus en plus nombreux. Avoir son serveur permet d'héberger soi-même son site web ou fournir des services, comme un serveur de jeux. L'administration de serveur sous Linux se fait presque exclusivement en ligne de commande à distance, à l'aide de SSH.

Se connecter à une console à distance

Jusqu'ici, vous avez utilisé Linux de la même façon que Windows : vous étiez **en face** de votre ordinateur. Vous étiez physiquement à côté de votre machine, vous avez par exemple appuyé sur le bouton « Power » pour l'allumer. Jusque-là, rien de nouveau.

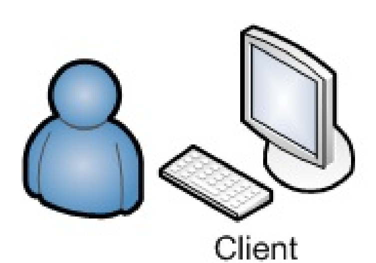
Pourtant, une des grandes forces de Linux est que l'on peut s'en servir même si l'on est à des centaines de kilomètres de la machine. Ce fonctionnement date de l'époque d'Unix où il était nécessaire d'administrer des machines à distance.
Aujourd'hui, si j'habite à Paris, je peux très bien contrôler un ordinateur sous Linux situé à Tokyo, au Japon, en même temps qu'un autre ordinateur situé au fin fond du Nevada, aux États-Unis. Je peux même ordonner à l'ordinateur de Tokyo d'envoyer un fichier à celui du Nevada.

Ce genre de manipulation est désormais possible grâce à l'internet, et cela se fait tous les jours. Les personnes qui s'emploient à gérer des machines Linux, souvent à distance, sont appelées **administrateurs système** (c'est un métier recherché !).
Heureusement qu'il n'est pas nécessaire d'être présent physiquement à côté de la machine pour travailler dessus ! Vous imaginez, devoir se payer un billet aller-retour pour Tokyo juste parce que l'on a besoin d'installer un programme sur un serveur…



Un **serveur** est un ordinateur qui reste allumé 24 h / 24, 7 j / 7. Cet ordinateur est semblable au vôtre (quoique souvent plus puissant et plus bruyant) : il possède un processeur, un ou plusieurs disques durs, etc.
Le principe d'un serveur est de rester allumé et connecté à l'internet tout le temps. Il offre des services. Par exemple, le Site du Zéro possède plusieurs serveurs chargés de vous envoyer les pages web du site à toute heure du jour et de la nuit. :-)

Le PC qui se connecte au serveur est appelé le **client**. Nous allons les représenter comme sur la figure suivante dans les prochains schémas.





Actuellement, votre petit PC chez vous n'est pas considéré comme un serveur… mais vous pouvez très facilement le transformer en serveur si vous le désirez, à condition d'installer les bons programmes et de les configurer correctement.
Et de le laisser allumé aussi, parce qu'un serveur éteint, c'est un serveur qui ne sert à rien.

Nous allons suivre ce plan pour découvrir SSH :

1. Pourquoi faut-il sécuriser les échanges ?
2. Comment fait SSH pour sécuriser les échanges ?
3. Comment utiliser SSH concrètement ?

De Telnet à SSH

Les protocoles

Pour communiquer entre eux en réseau, deux ordinateurs doivent utiliser le même **protocole**. C'est un peu comme une langue : pour parler à quelqu'un, vous devez parler la même langue que lui, sinon vous ne vous comprendrez pas.

Il existe de très nombreux protocoles pour que les ordinateurs puissent communiquer entre eux. Il y en a un que vous avez forcément vu, c'est le HTTP (*HyperText Transfer Protocol*). Si, si, regardez par exemple l'adresse du Site du Zéro. Le préfixe `http` signifie que vous communiquez avec les serveurs du Site du Zéro à l'aide du protocole HTTP. C'est le protocole utilisé sur le web pour s'échanger des pages web.

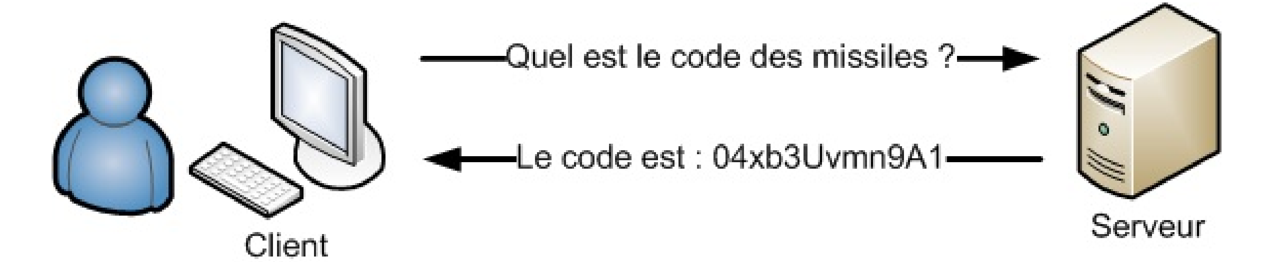
Mais il existe bien d'autres protocoles ! Par exemple le FTP (*File Transfer Protocol*, protocole de transfert de fichiers), l'IMAP (*Internet Message Access Protocol*, utilisé pour s'échanger des e-mails), etc.

Le protocole Telnet : simple mais dangereux

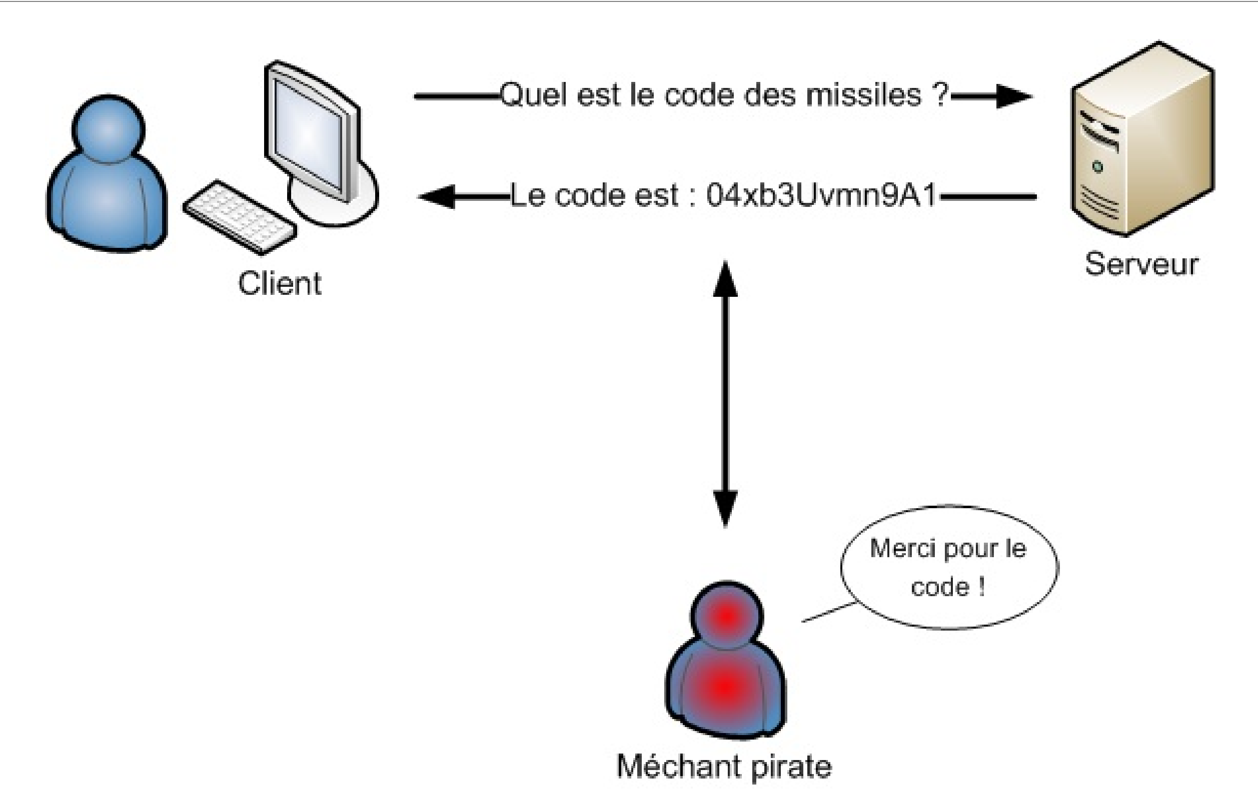
Un protocole très simple, très basique, a été créé dans les années 80 : c'est **Telnet**. Il sert juste à échanger des messages simples d'une machine à une autre.

En théorie donc, on peut communiquer avec un serveur à l'aide du protocole Telnet. Le problème de ce protocole… c'est justement qu'il est trop simple : les données sont transférées en clair sur le réseau. Il n'y a aucun cryptage.

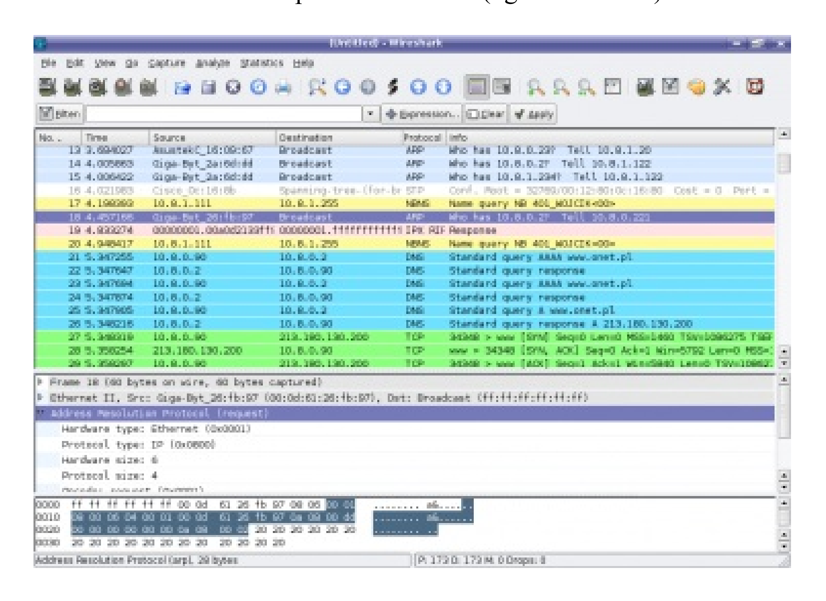
Voici ce qui pourrait se passer. Je force le trait, mais c'est pour vous donner une idée. Imaginez qu'un PC militaire demande à un serveur de l'armée le code de lancement de missiles (nucléaires, soyons fous), comme sur la figure suivante.



Après tout, il n'y a rien de choquant. Le message n'est envoyé qu'au client qui l'a demandé.
Mais en fait, un pirate aurait la possibilité d'« **écouter** » ce qui se passe sur le réseau, et donc d'intercepter les données en chemin (figure suivante).

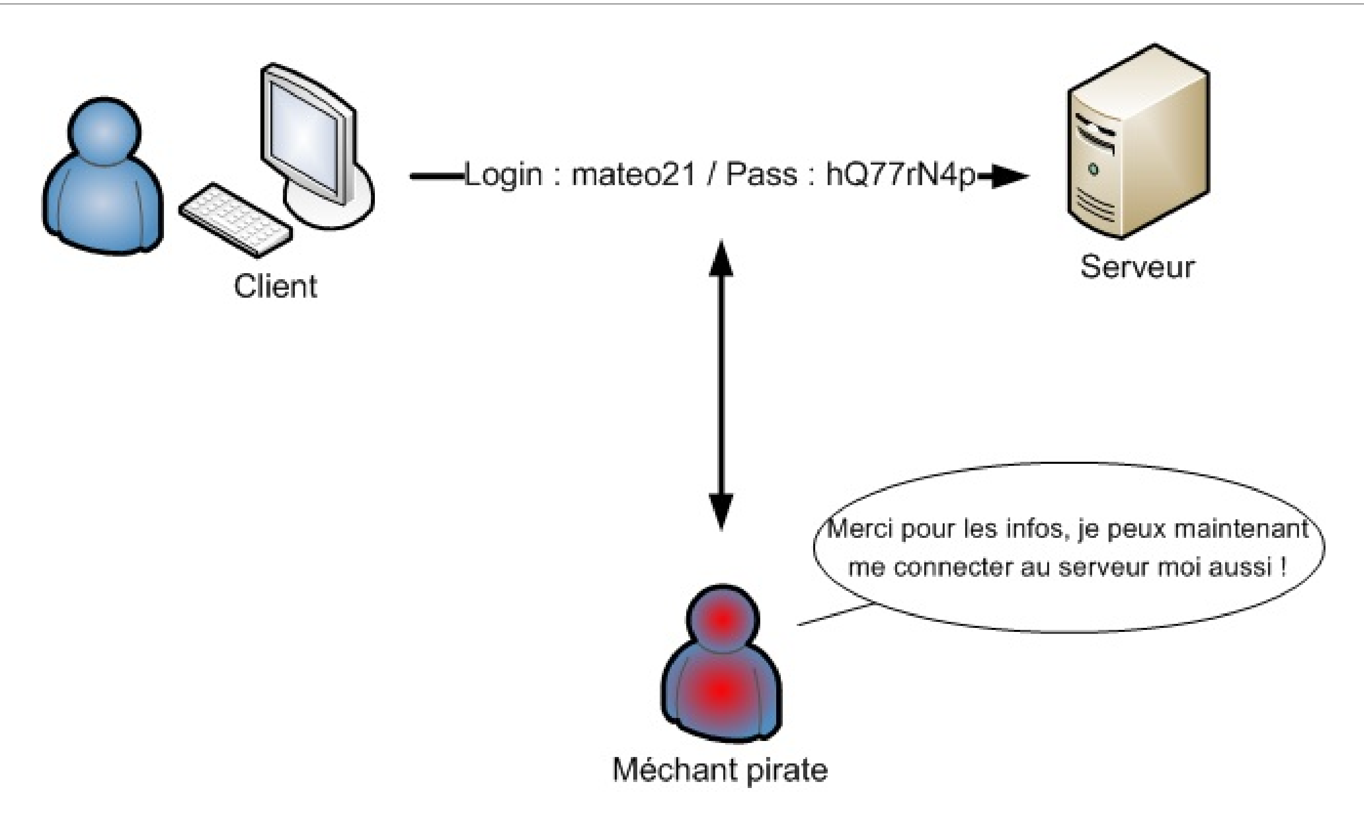


Vous pouvez difficilement empêcher que quelqu'un intercepte les données. Intercepter les données peut être compliqué à réaliser, mais possible. Sachez qu'il existe par exemple des programmes comme Wireshark capables d'écouter ce qui se passe — notamment sur un réseau local — et donc d'intercepter les données (figure suivante).



Euh… attends, là : moi, je veux juste me connecter à distance à ma machine ou à un serveur pour avoir l'accès à la console. Je ne vais pas échanger de code de lancement de missiles nucléaires ! Je vois pas en quoi c'est un problème si quelqu'un sait que je suis en train de faire un `grep` sur ma machine, par exemple…

Ça ne vous dérange pas que l'on vous espionne ? Soit.
Mais quand vous allez vous connecter au serveur, vous allez donner votre login et votre mot de passe. Rien que ça, c'est dangereux (figure suivante). Il ne faut pas que le login et le *pass* apparaissent en clair sur le réseau !



Rien que pour cela, il faut que les données soient cryptées. Vous ne voulez pas que quelqu'un récupère votre mot de passe tout de même !

Le protocole SSH : la solution pour sécuriser les données

Comme on ne peut pas complètement empêcher quelqu'un d'intercepter les données qui transitent sur l'internet, il faut trouver un moyen pour que le client et le serveur communiquent de manière sécurisée. Le cryptage sert précisément à ça : si le pirate récupère le mot de passe crypté, il ne peut rien en faire.

Mais tout cela est plus compliqué que ça en a l'air. Comment crypter les données ?

Comment sont cryptés les échanges avec SSH ?

SSH est un protocole assez complexe, mais il est vraiment intéressant de savoir comment il fonctionne. Plutôt que de l'utiliser bêtement, je vous propose de vous expliquer dans les grandes lignes son mode de fonctionnement.

Nous allons ici nous intéresser aux deux questions suivantes.

1. Quelles sont les différentes méthodes de cryptage qui existent ?
2. Comment SSH utilise-t-il ces méthodes de cryptage pour garantir la sécurité ?

Quelles sont les différentes méthodes de cryptage ?

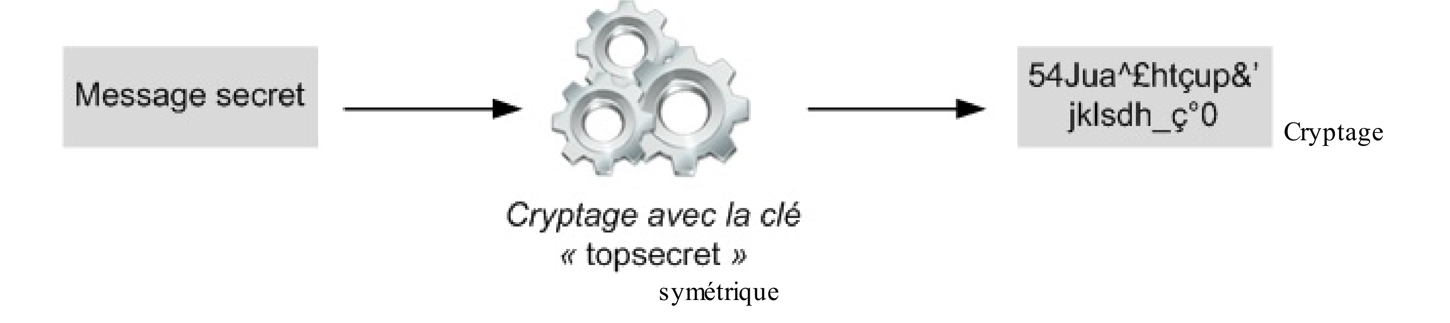
Il existe des tonnes d'algorithmes de cryptage. Je ne vais pas tous vous les présenter : cela demanderait trop de notions mathématiques, on pourrait y consacrer 30 chapitres et on n'aurait pas tout vu.

Si l'on ne peut pas connaître tous les algorithmes de cryptage, il faut par contre savoir que l'on peut les classer en deux catégories : les cryptages *symétriques* et les cryptages *asymétriques*.

Le cryptage symétrique

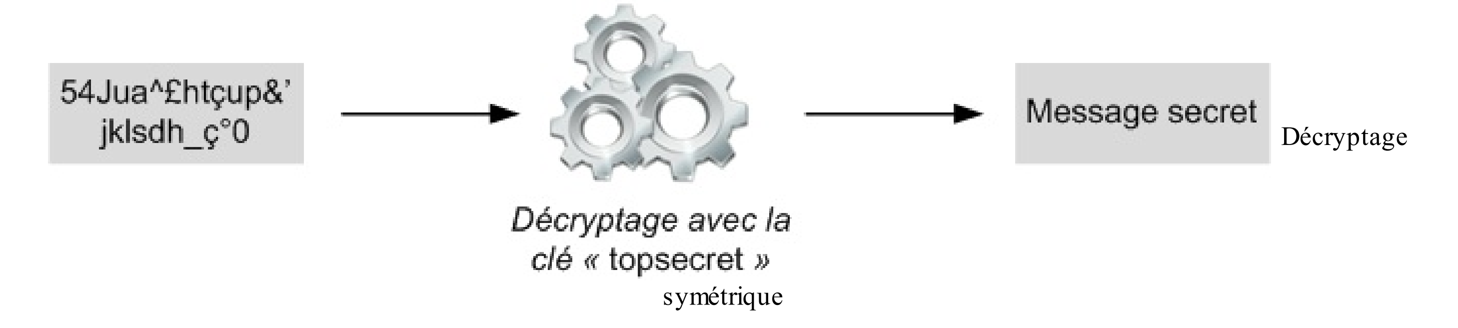
C'est la méthode de cryptage la plus simple. Cela ne veut pas dire qu'elle n'est pas robuste (il existe des cryptages symétriques très sûrs). Cela veut plutôt dire que le fonctionnement est simple à comprendre. :-)

Avec cette méthode, on utilise une clé (un mot de passe secret) pour crypter un message. Par exemple, imaginons que cette clé soit `topsecret` (figure suivante).



Cryptage symétrique

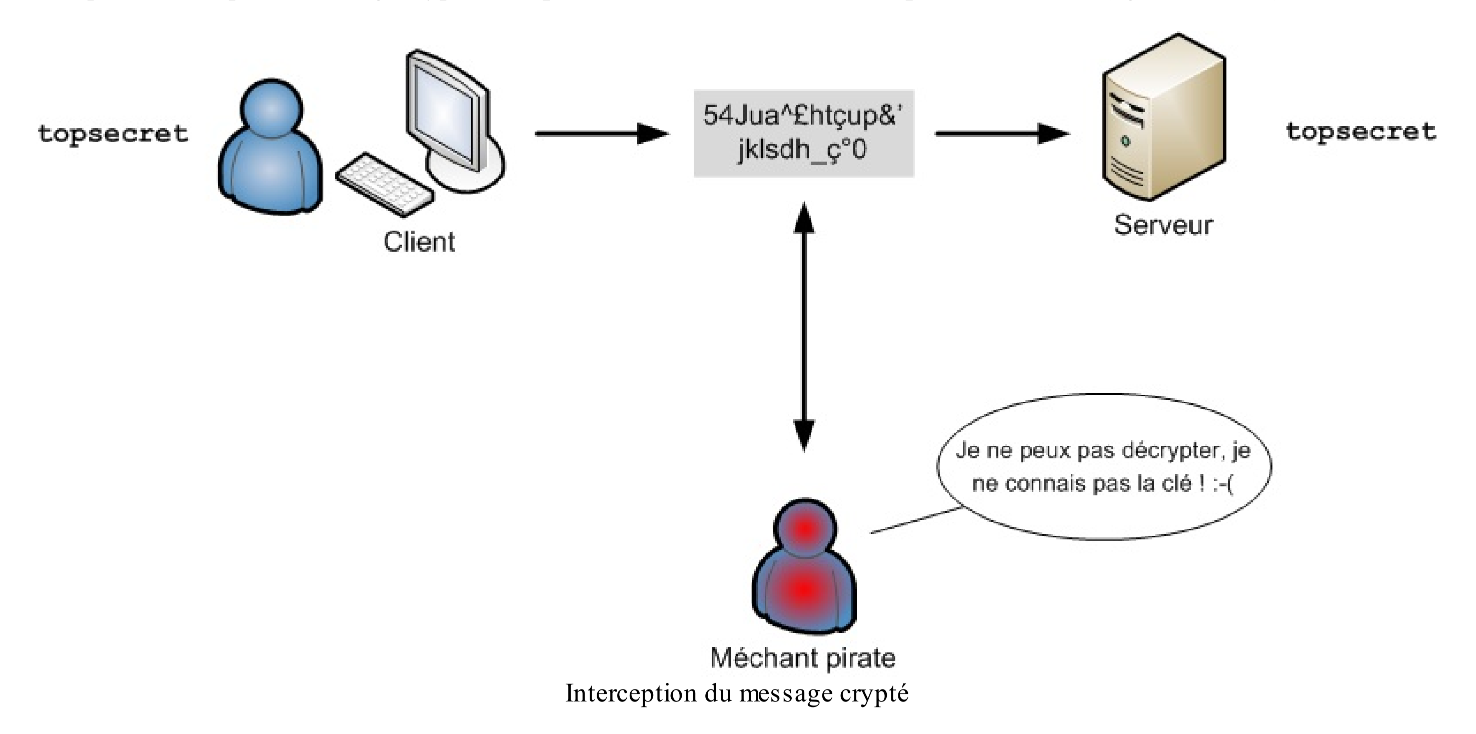
Pour décrypter ensuite le message, on utilise cette même clé (figure suivante)…



Décryptage symétrique

Il faut donc que la personne qui crypte et celle qui décrypte connaissent toutes deux cette clé qui sert à crypter et décrypter.

Si le pirate intercepte un message crypté, **il ne peut rien en faire s'il ne connaît pas la clé secrète** (figure suivante) !

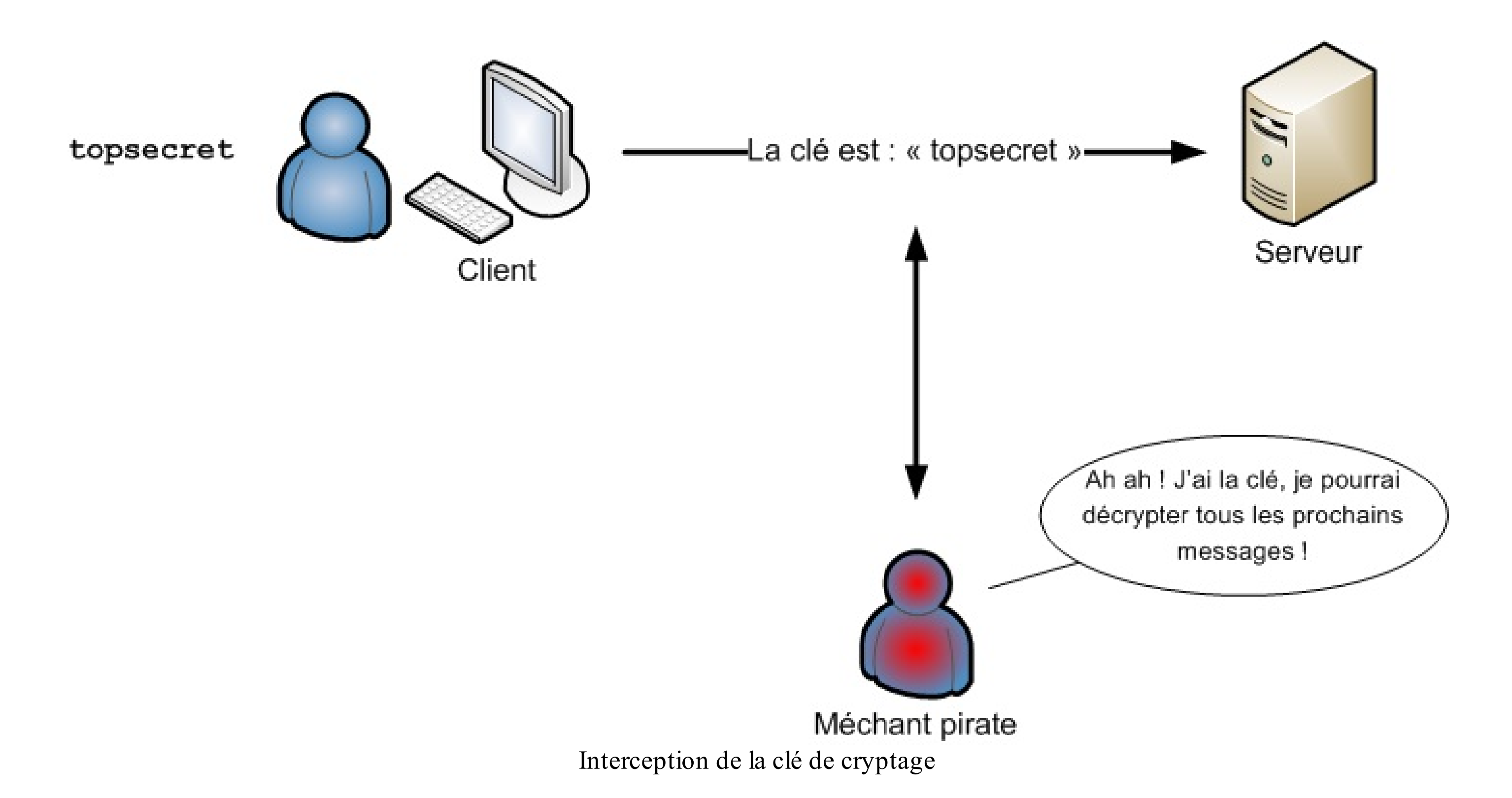


Interception du message crypté

Ah… c'est bien, ça ! Mais il faut que le client et le serveur connaissent tous les deux la clé de cryptage.
Il faut donc que le client envoie d'abord au serveur la clé pour que celui-ci puisse décrypter ses futurs messages…

Très bonne remarque : je vois que vous suivez, c'est bien. ;-)

En effet, pour que le schéma que l'on vient de voir puisse fonctionner, il faut que le client et le serveur se soient transmis auparavant la clé magique qui sert à crypter et décrypter. Mais comment font-ils pour se l'échanger ? S'ils l'envoient en clair, le pirate va pouvoir l'intercepter et sera ensuite capable de décrypter tous les messages cryptés qui passeront sur le réseau (voyez la figure suivante) !



Interception de la clé de cryptage

Le cryptage symétrique est donc puissant, mais il a un gros défaut : il faut communiquer « discrètement » la clé de cryptage… mais c'est impossible : il faut bien envoyer la clé en clair au début !

…
À moins de… non…
Et pourquoi pas ? Si l'on cryptait la clé de cryptage lors de son envoi ? :-p

Pour crypter la clé de cryptage symétrique, on va utiliser une autre méthode : le cryptage asymétrique. Avec cette autre méthode, on ne risque pas de connaître à nouveau le problème que l'on vient de rencontrer.

Le cryptage asymétrique

Le cryptage symétrique utilise une seule clé pour crypter et décrypter.
Le cryptage asymétrique, lui, utilise une clé pour crypter, et une autre pour décrypter.

Il y a donc deux clés :

- une clé dite « **publique** » qui sert à **crypter** ;
- une clé dite « **privée** » qui sert à **décrypter**.

La clé publique ne sert qu'à crypter. Avec ce type d'algorithme, on ne peut décrypter un message que si l'on connaît la clé privée.

On demande à l'ordinateur de générer une paire de clés : une privée et une publique. Elles vont ensemble.
Ne me demandez pas comment il les génère ni pourquoi elles vont ensemble, c'est trop compliqué à expliquer ici. Admettez simplement que l'ordinateur est capable de générer aléatoirement un couple de clés qui vont ensemble.

Prenons un exemple et imaginons que :

- la clé publique soit **74A48vXX** ;
- la clé privée soit **99o0pn9**.

Pour crypter, on utilise la clé publique, comme sur la figure suivante.



Cryptage asymétrique

Pour décrypter, la clé publique ne fonctionne pas. Il faut obligatoirement utiliser la clé privée (figure suivante).



Décryptage asymétrique

Voilà pourquoi on dit que c'est un cryptage **asymétrique** : il faut deux clés différentes. L'une d'elles permet de crypter le message, l'autre de le décrypter. Il n'y a pas d'autre moyen.

La clé publique peut être transmise en clair sur le réseau (elle est « publique »). Ce n'est pas grave si un pirate l'intercepte. En revanche, la clé privée — qui permet donc de décrypter — doit rester secrète.



L'algorithme de cryptage asymétrique le plus connu s'appelle RSA. Si vous voulez savoir comment RSA fonctionne et pourquoi il faut une clé différente pour crypter et pour décrypter, il existe sur le Site du Zéro un tutoriel sur RSA. Je vous préviens : il faut aimer les maths.

La création d'un tunnel sécurisé avec SSH

SSH combine cryptage asymétrique et cryptage symétrique

SSH utilise les deux cryptages : asymétrique et symétrique. Cela fonctionne dans cet ordre.

1. On utilise d'abord le cryptage asymétrique pour s'échanger discrètement une clé secrète de cryptage symétrique.
2. Ensuite, on utilise tout le temps la clé de cryptage symétrique pour crypter les échanges.



Pourquoi ne pas utiliser uniquement du cryptage asymétrique tout le temps ?

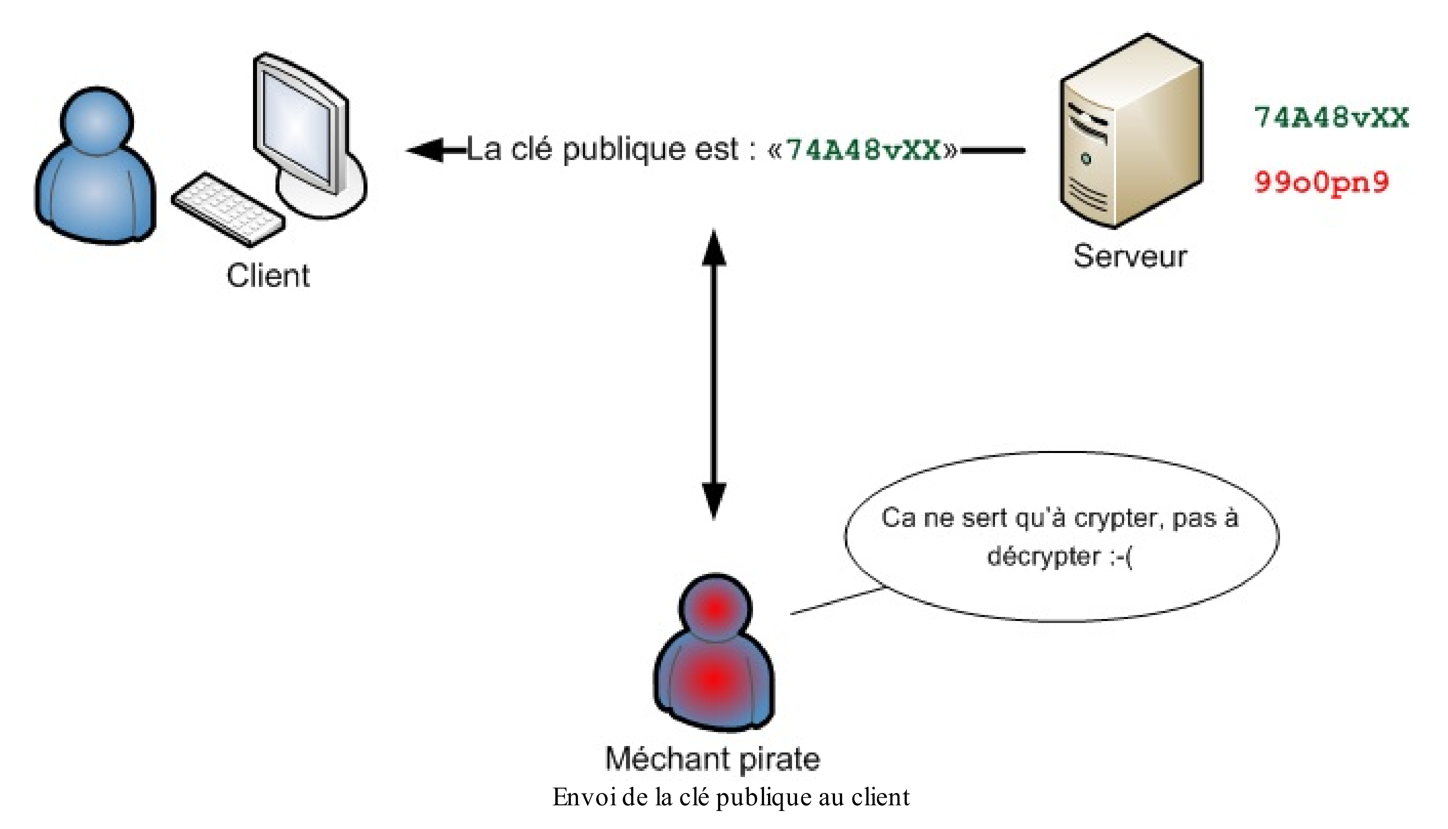
Ce serait possible mais il y a un défaut : le cryptage asymétrique demande beaucoup trop de ressources au processeur. Le cryptage asymétrique est 100 à 1 000 fois plus lent que le cryptage symétrique !
Les ordinateurs s'échangent donc la clé de cryptage symétrique de manière sécurisée (grâce au cryptage asymétrique) et peuvent ensuite communiquer plus rapidement en utilisant en permanence le cryptage symétrique.

Le cryptage asymétrique est donc utilisé seulement au début de la communication, afin que les ordinateurs s'échangent la clé de cryptage symétrique de manière sécurisée. Ensuite, ils ne communiquent que par cryptage symétrique.

Les étapes de la création d'un canal sécurisé avec SSH en images

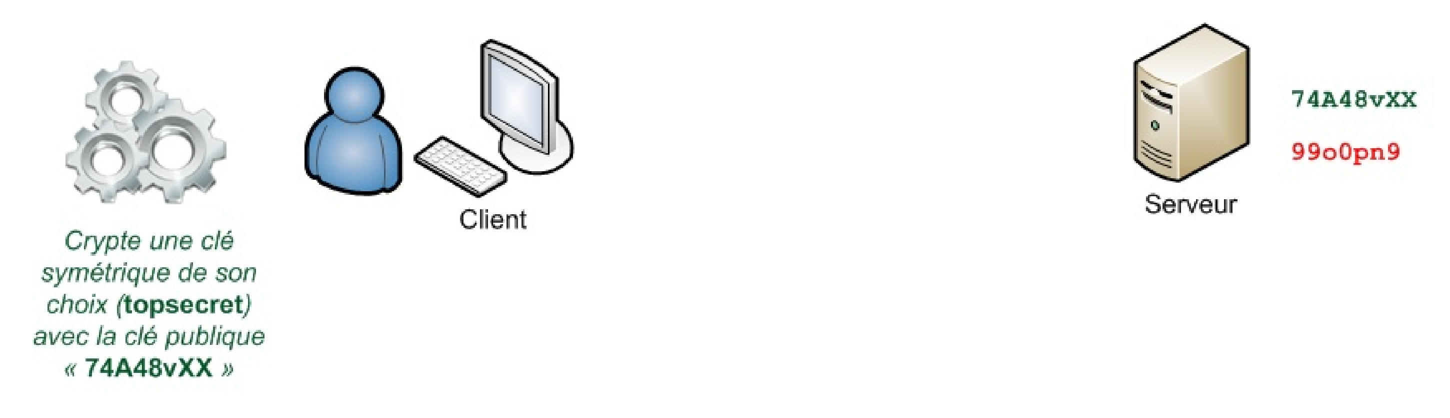
Je résume en images. On veut s'échanger une clé de cryptage symétrique, mais on ne peut pas le faire en clair sinon le pirate peut l'intercepter. On va donc crypter la clé grâce au cryptage asymétrique.

Le serveur envoie la clé publique en clair au client pour qu'il puisse crypter (figure suivante).



Envoi de la clé publique au client

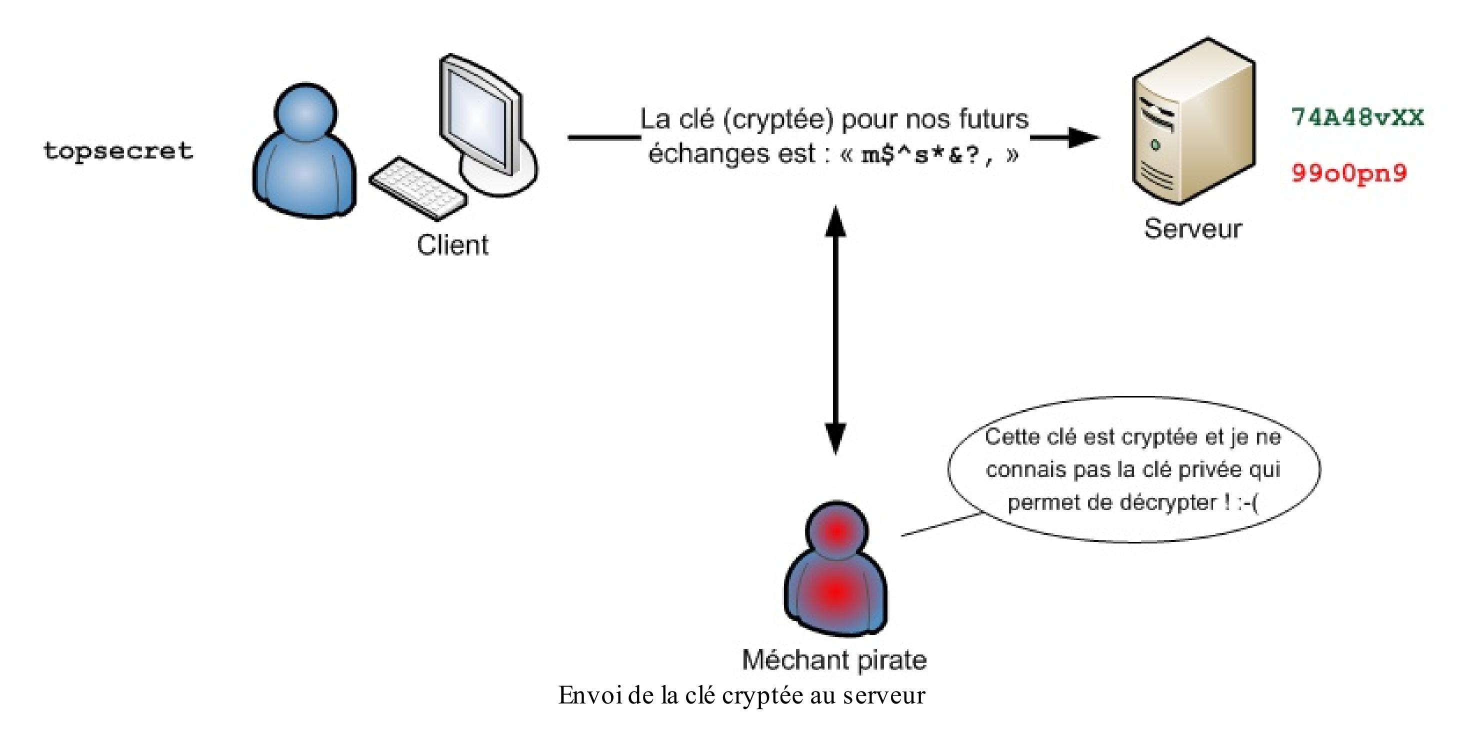
Le client génère une clé de cryptage symétrique (par exemple `topsecret`) qu'il crypte grâce à la clé publique qu'il a reçue (figure suivante).





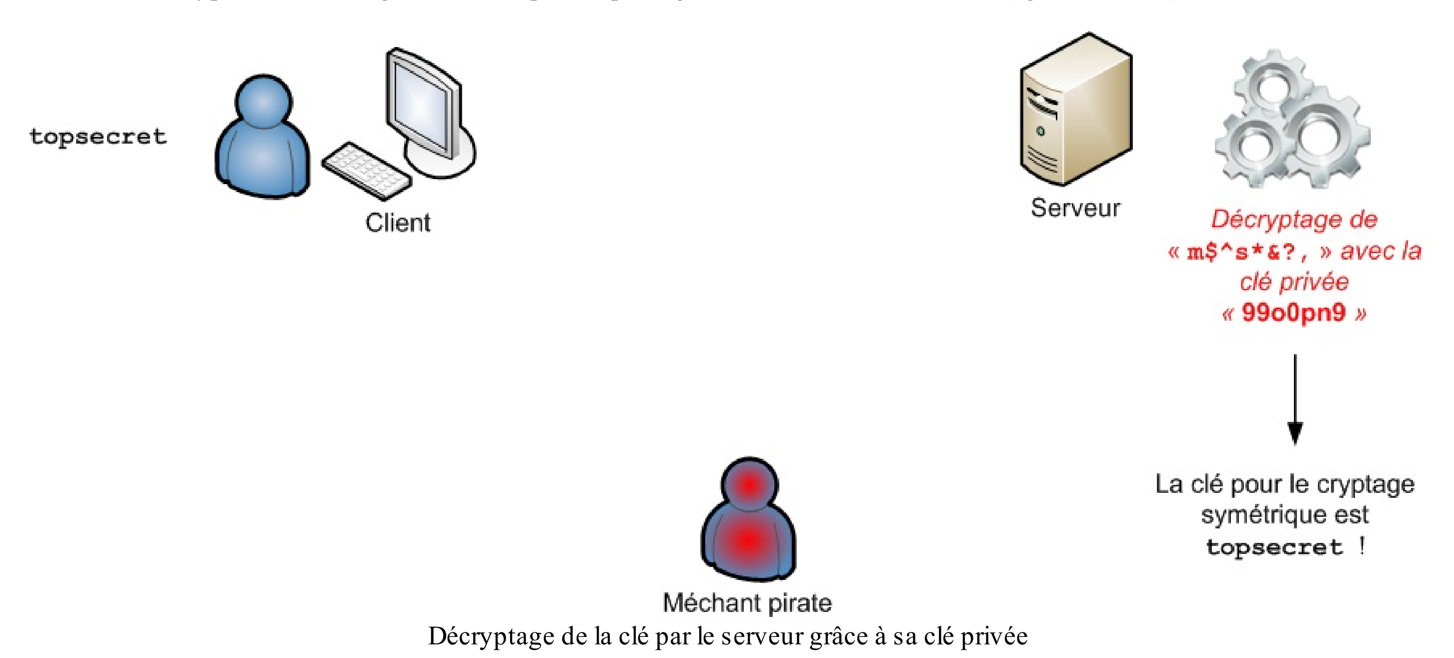
Création de clé symétrique et cryptage par le client

Le client envoie la clé symétrique cryptée au serveur. Le pirate peut l'intercepter, mais ne peut pas la décrypter car il faut pour cela la clé privée, connue seulement du serveur (figure suivante).



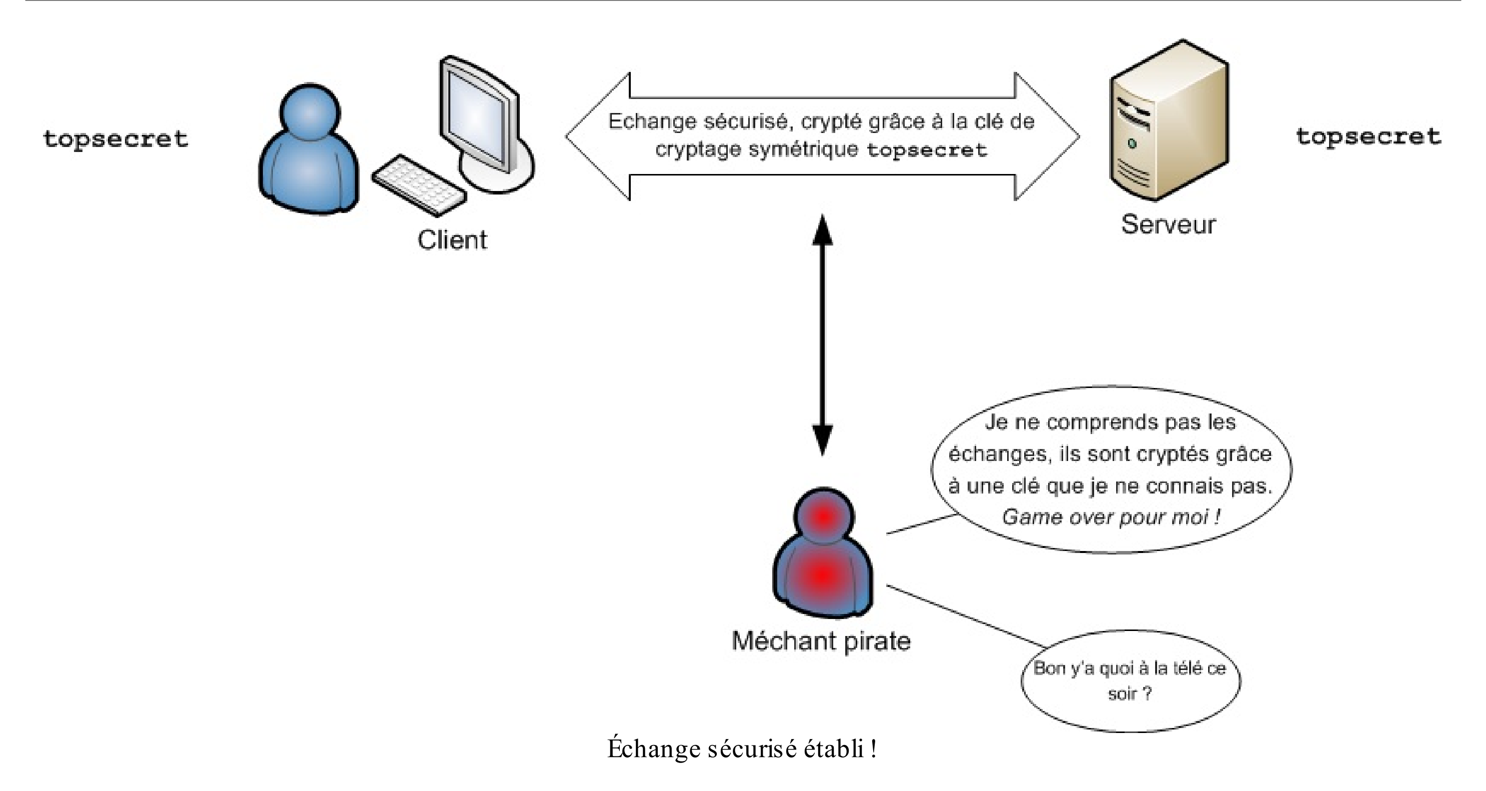
Envoi de la clé cryptée au serveur

Le serveur décrypte la clé reçue grâce à sa clé privée qu'il a gardée bien au chaud chez lui (figure suivante).



Décryptage de la clé par le serveur grâce à sa clé privée

Le client et le serveur connaissent maintenant tous les deux la clé symétrique `topsecret`, et à aucun moment ils ne l'ont échangée en clair sur le réseau !
Ils peuvent donc s'envoyer des messages cryptés de manière symétrique en toute tranquillité. Ce cryptage est plus rapide et tout aussi sûr que le cryptage asymétrique car le pirate ne connaît pas la clé (figure suivante) !



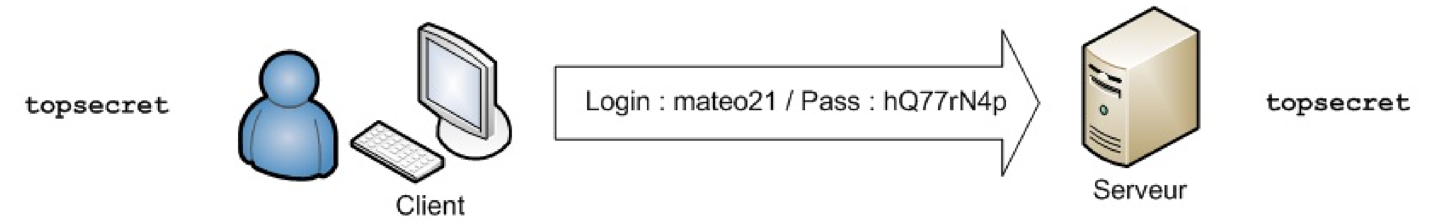
Échange sécurisé établi !

Voilà comment SSH fonctionne pour créer un canal d'échange sécurisé. Tout est crypté grâce à la clé symétrique que le client et le serveur se sont **astucieusement** communiquée.



Maintenant qu'ils discutent de manière sécurisée, que font le client et le serveur ?

Eh bien **seulement** maintenant, le client peut se connecter au serveur : il peut donner son login et son mot de passe sans craindre de se les faire voler par le pirate (figure suivante) ! ;-)



Le client peut envoyer son login et son mot de passe de manière sécurisée



Faut-il savoir tout cela pour utiliser SSH ?

Non. En fait, tout se fait automatiquement. Vous allez juste avoir à entrer un login et un mot de passe pour vous connecter à votre machine à distance.
Mais j'estime que c'était l'occasion idéale de vous expliquer comment fonctionne le protocole SSH. Ce système est utilisé partout dans le monde ! Plus personne n'envisage de se connecter en Telnet aujourd'hui.

Se connecter avec SSH et PuTTY

Assez de théorie, passons à la pratique ! Vous allez voir, ça sera beaucoup plus simple car les ordinateurs effectuent les cryptages entre eux sans nous demander d'intervenir… et c'est tant mieux. ;-)

À partir de maintenant, de deux choses l'une.

- Soit **vous louez déjà un serveur dédié** (ce qui devrait être le cas d'une minorité d'entre vous). Celui-ci est déjà configuré comme serveur SSH, vous n'avez donc rien à faire pour le « transformer » en serveur.

Si vous voulez louer un serveur dédié, sachez qu'il existe de très nombreux hébergeurs qui en proposent, comme



par exemple *OVH*. Comme vous pourrez le constater, ça coûte cher (en même temps, c'est un ordinateur à part entière que vous louez !).
Sachez qu'il existe aussi des serveurs *low cost*, moins chers (moins puissants mais ils peuvent suffire), comme *Kimsufi* et *Dedibox*.
Je vous recommande d'attendre un peu avant de louer un serveur dédié : cela représente un gros investissement et il vaut mieux être sûr d'en avoir vraiment besoin.

- Soit **vous n'avez pas de serveur dédié**, ce qui, je suppose, est le cas de la plupart d'entre vous. Dans ce cas, nous allons voir tout de suite comment transformer votre PC en serveur.

Transformer sa machine en serveur

Cette étape vous concerne si vous voulez transformer votre PC en serveur. Par exemple, si vous voulez accéder à votre PC depuis un autre lieu (et donc suivre le reste de ce chapitre), vous devez le transformer en serveur au préalable.

Il faut tout simplement installer le paquet `openssh-server` :

Code : Console

```
sudo apt-get install openssh-server
```

Lors de l'installation, vous devriez voir certaines étapes intéressantes s'effectuer automatiquement :

Code : Console

```
Creating SSH2 RSA key; this may take some time ...
Creating SSH2 DSA key; this may take some time ...
 * Restarting OpenBSD Secure Shell server sshd                          [ OK ]
```

RSA et DSA sont deux algorithmes de cryptage asymétrique. Comme je vous l'ai dit plus tôt, SSH peut travailler avec plusieurs algorithmes de cryptage différents.
Ce que vous voyez là est l'étape de création d'une clé publique et d'une clé privée pour chacun des deux algorithmes (RSA et DSA).

Ensuite, le programme de serveur SSH (appelé `sshd`) est lancé.

Normalement, le serveur SSH sera lancé à chaque démarrage. Si ce n'est pas le cas, vous pouvez le lancer à tout moment avec la commande suivante :

Code : Console

```
sudo /etc/init.d/ssh start
```

Et vous pouvez l'arrêter avec cette commande :

Code : Console

```
sudo /etc/init.d/ssh stop
```

Logiquement, vous ne devriez pas avoir besoin de configurer quoi que ce soit, mais sachez au besoin que le fichier de configuration se trouve dans `/etc/ssh/ssh_config`. Il faudra recharger SSH avec la commande `sudo /etc/init.d/ssh reload` pour que les changements soient pris en compte.

Voilà : votre machine est désormais un serveur SSH ! Vous pouvez vous y connecter depuis n'importe quel ordinateur sous Linux ou sous Windows dans le monde (pour peu que vous ne soyez pas derrière un pare-feu).

Nous commencerons dans un premier temps par voir comment accéder à votre PC à distance depuis une machine Linux.

Se connecter via SSH à partir d'une machine Linux

Toutes les machines équipées de Linux proposent la commande `ssh` qui permet de se connecter à distance à une autre machine.



À partir d'ici, je suppose que vous avez installé `openssh-server` et que votre machine est allumée. L'idéal serait d'aller chez un ami qui a Linux (ou d'utiliser de chez vous un autre PC équipé de Linux).

Ouvrez une console sur le second PC et utilisez la commande `ssh` comme ceci :

Code : Console

```
ssh login@ip
```

Il faut remplacer `login` par votre login (mateo21, dans mon cas) et `ip` par l'adresse IP de votre ordinateur.



Si vous vous connectez depuis chez un ami, il vous faut entrer l'IP internet de votre PC que vous pouvez obtenir en allant sur www.whatismyip.com par exemple.
Si vous vous connectez depuis un autre PC chez vous (sur le même réseau local), il vous faut entrer l'IP locale que vous devriez voir en tapant la commande `ifconfig` (par exemple 192.168.0.3).
Si **vraiment** vous n'avez ni ami sous Linux ni second PC dans la maison, vous pouvez simuler une connexion réseau en vous connectant depuis votre PC à votre PC. Utilisez pour cela l'IP 127.0.0.1 (ou le mot `localhost`), ça marche toujours.

Si je suis chez un ami et que l'IP internet de mon ordinateur est 87.112.13.165, je vais taper :

Code : Console

```
ssh mateo21@87.112.13.165
```

Si, faute de mieux, vous voulez tester en vous connectant chez vous depuis chez vous, vous pouvez taper :

Code : Console

```
ssh mateo21@localhost
```

Cette seconde méthode marche toujours, mais c'est moins impressionnant parce que vous ne faites que simuler une connexion réseau.

Normalement, le serveur devrait répondre au bout d'un moment et vous devriez voir quelque chose comme ce qui suit :

Code : Console

```
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is 49:d9:2d:2a:df:fd:80:ab:e9:eb:59:37:58:34:de:f7.
Are you sure you want to continue connecting (yes/no)?
```



Si vous n'avez pas de réponse du serveur, vérifiez que vous ne vous êtes pas trompés d'IP. Vérifiez aussi que le port 22 n'est pas bloqué par un pare-feu, car c'est celui utilisé par SSH par défaut.
Si le serveur tourne sur un autre port, il faudra préciser le numéro de ce port comme ceci :
`ssh mateo21@87.112.13.165 -p 12451` (si le serveur fonctionne sur le port 12451 au lieu du port 22).

Que se passe-t-il ? On vous dit que le *fingerprint* (empreinte) du serveur est `49:d9:2d:2a:df:fd:80:ab:e9:eb:59:37:58:34:de:f7`. C'est un numéro unique qui vous permet d'identifier le serveur. Si demain quelqu'un essaie de se faire passer pour le serveur, le *fingerprint* changera forcément et vous saurez qu'il se passe alors quelque chose d'anormal. Ne vous inquiétez pas, SSH vous avertira de manière très claire si cela arrive.

En attendant, tapez « yes » pour confirmer que c'est bien le serveur auquel vous voulez vous connecter. Le serveur et le client vont alors s'échanger une clé de cryptage, comme je vous l'ai expliqué un peu plus tôt. Normalement, le serveur devrait vous demander au bout de quelques secondes votre mot de passe :

Code : Console

```
mateo21@localhost's password:
```

Vous pouvez l'entrer en toute sécurité, la communication est cryptée. ;-)
Si vous entrez le bon mot de passe, la console du PC de votre ami (ou votre propre console) devrait vous afficher un message de bienvenue puis un **prompt** qui correspond à la console de votre PC. Bravo, vous êtes connectés !

Code : Console

```
mateo21@mateo21-desktop:~$
```

Si aucune erreur ne s'affiche, c'est que vous êtes bien connectés et que vous travaillez désormais à distance sur votre machine !
Vous pouvez effectuer toutes les opérations que vous voulez comme si vous étiez chez vous.
Essayez de parcourir les dossiers pour voir que ce sont bien les vôtres, et amusez-vous (pourquoi pas) à créer un fichier (avec Nano). Lorsque vous reviendrez sur votre PC, vous l'y retrouverez.

Vous pouvez aussi commander l'exécution d'un programme, d'une recherche, etc. Vous savez déjà comment lancer un programme en tâche de fond pour qu'il continue de s'exécuter même quand vous n'êtes pas connectés à la machine.
Vous vous souvenez de `nohup` et de `screen` ?

Pour vous déconnecter, tapez `logout` ou son équivalent : la combinaison de touches `Ctrl` + `D`.

Se connecter via SSH à partir d'une machine Windows

Si vous voulez avoir accès à la console de votre machine Linux mais que vous n'avez pas d'autre machine Linux sous la main, pas de panique ! Il existe des programmes pour Windows faits pour cela. Le plus connu d'entre eux – celui que j'utilise personnellement – s'appelle **PuTTY**.

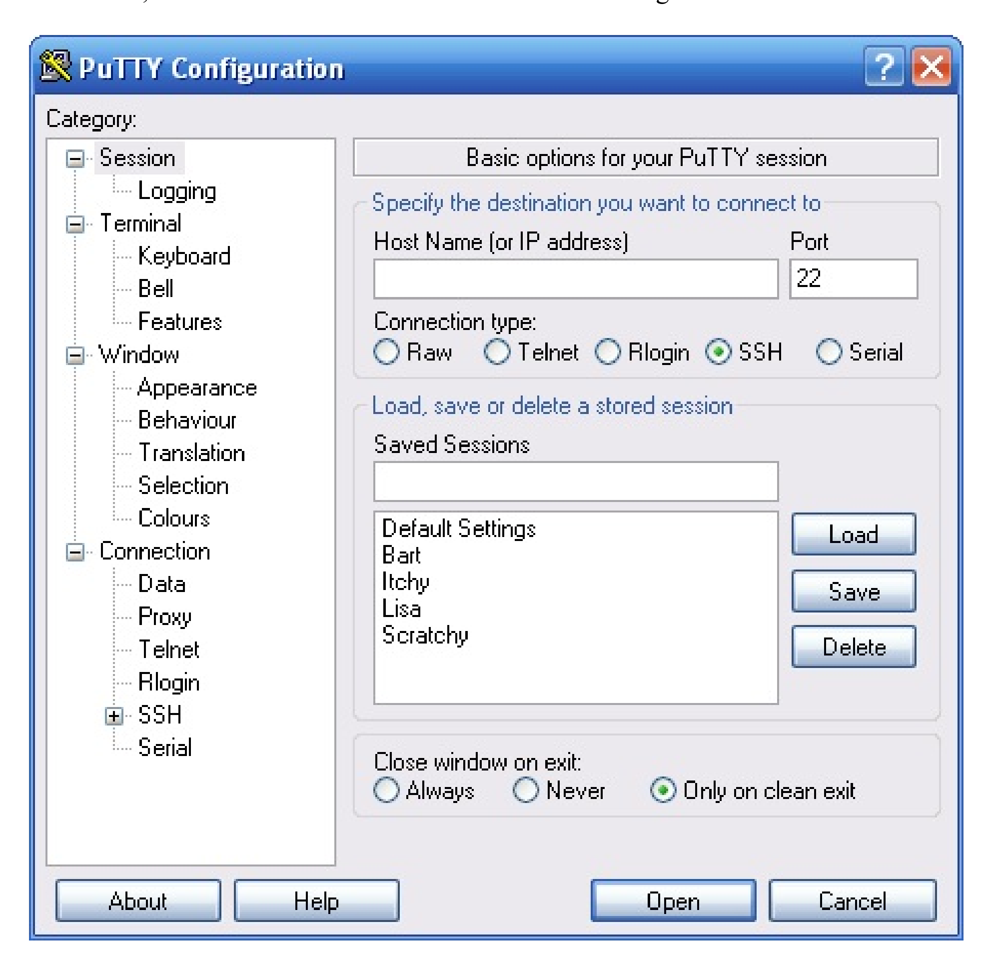
Vous pouvez télécharger PuTTY depuis son site officiel.

Je sais : vous devez vous dire que ce n'est pas très clair et que vous ne voulez pas chercher sur quel lien cliquer sur cette page. Repérez la section « Binaries ». C'est un tableau. Vous avez le choix entre :

- cliquer sur `putty.exe` pour télécharger le programme principal. Il ne nécessite pas d'installation ;
- cliquer sur le programme d'installation (par exemple `putty-0.60-installer.exe`). Celui-ci installera PuTTY et d'autres utilitaires dont vous aurez besoin dans quelques minutes.

`putty.exe` suffit, mais je vous recommande de prendre le package complet en récupérant le programme d'installation.

Une fois que c'est fait et installé, lancez PuTTY. Une fenêtre comme celle de la figure suivante devrait s'afficher.

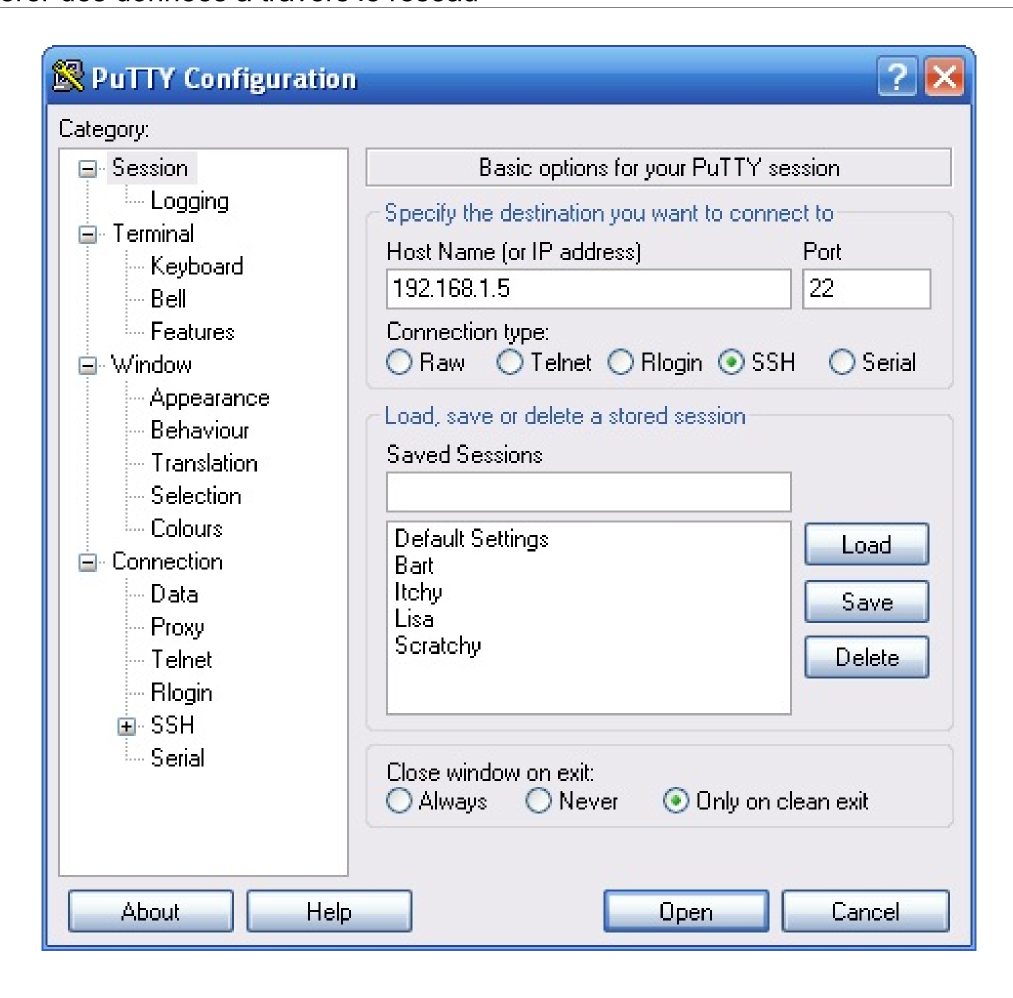


Il y a beaucoup de pages d'options, comme vous pouvez le voir au niveau de la section « Category » sur le côté. Pour le moment, pas de panique : vous avez juste besoin de remplir le champ en haut « Host Name (or IP address) ». Entrez-y l'adresse IP de votre ordinateur sous Linux.



J'ai donné quelques explications à propos de l'adresse IP un peu plus haut, lorsque j'ai parlé de la connexion SSH depuis Linux. Lisez donc les paragraphes précédents si vous voulez plus d'informations à ce sujet.

Dans mon cas, je vais entrer l'adresse IP de mon PC sous Linux situé sur mon réseau local (192.168.1.5 – figure suivante).



Vous pouvez changer le numéro du port si ce n'est pas 22, mais normalement c'est 22 par défaut.

Ensuite, vous n'avez plus qu'à cliquer sur le bouton « Open » tout en bas pour lancer la connexion. Rien d'autre !



Si vous voulez sauvegarder l'IP et les paramètres pour ne pas devoir les retaper à chaque fois, donnez un nom à cette connexion (par exemple, le nom de votre ordinateur) dans le champ sous « Saved Sessions », puis appuyez sur le bouton « Save ». La prochaine fois, vous n'aurez qu'à double-cliquer sur le nom de votre PC dans la liste pour vous y connecter directement.

La première fois que vous vous connectez à votre serveur, PuTTY devrait vous demander une confirmation comme sur la figure suivante.



C'est la même chose que sous Linux : on vous donne l'empreinte (*fingerprint*) de votre serveur. Vous devez confirmer que c'est bien chez lui que vous voulez vous connecter. Cliquez sur « Oui » pour confirmer.

À l'avenir, on ne vous reposera plus la question. Par contre, si le *fingerprint* change, un gros message d'avertissement s'affichera. Cela signifiera soit que le serveur a été réinstallé, soit que quelqu'un est en train de se faire passer pour le serveur (c'est ce que l'on appelle une attaque *man-in-the-middle*). Cela ne devrait fort heureusement pas vous arriver, du moins je l'espère. :-)

Le serveur vous demande alors le login et le mot de passe (figure suivante).

```
192.168.1.5 - PuTTY
login as: mateo21
mateo21@192.168.1.5's password:
```



Rappelez-vous qu'il est normal que les caractères ne s'affichent pas quand vous tapez votre mot de passe. Il n'y a même pas d'étoiles pour des raisons de sécurité, afin que quelqu'un ne soit pas tenté de compter le nombre de caractères en regardant derrière votre épaule !

Si tout est bon, vous devriez être connectés à votre machine (figure suivante) !

```
mateo21@mateo21-laptop: ~
login as: mateo21
mateo21@192.168.1.5's password:
Linux mateo21-laptop 2.6.27-7-generic #1 SMP Tue Nov 4 19:33:20 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Mon Nov 17 14:50:35 2008 from localhost
mateo21@mateo21-desktop:~$
```

Et voilà, vous êtes chez vous ! Vous pouvez faire ce qui vous chante : lire vos fichiers, écrire des fichiers, lancer une recherche, exécuter un programme… bref, vous êtes chez vous.

Pour vous déconnecter, tapez `logout` ou son équivalent, la combinaison de touches `Ctrl` + `D`.

L'identification automatique par clé

Il y a plusieurs façons de s'authentifier sur le serveur, pour qu'il sache que c'est bien vous. Les deux plus utilisées sont :

- l'authentification par mot de passe ;
- l'authentification par clés publique et privée du **client**.

Pour le moment, nous avons vu uniquement l'authentification par mot de passe (le serveur vous demandait votre mot de passe). Il est possible d'éviter que l'on vous le demande à chaque fois grâce à une authentification spéciale par clé. Cette méthode d'authentification est plus complexe à mettre en place, mais elle est ensuite plus pratique.



Avec cette nouvelle méthode d'authentification, c'est le client qui va générer une clé publique et une clé privée. Les rôles sont un peu inversés.
L'avantage, c'est que l'on ne vous demandera pas votre mot de passe à chaque fois pour vous connecter. Si vous vous connectez très régulièrement à un serveur, c'est vraiment utile. Si vous faites bien les choses, cette méthode est tout aussi sûre que l'authentification par mot de passe.

Je vais, là encore, distinguer les deux cas :

- vous essayez de vous connecter depuis une machine Linux ;
- vous essayez de vous connecter depuis une machine Windows (avec PuTTY).

Authentification par clé depuis Linux

Pour mettre en marche ce mode d'authentification, nous allons d'abord devoir effectuer des opérations sur la machine du client, puis nous enverrons le résultat au serveur.

Opérations sur la machine du client

Il faut tout d'abord vous rendre sur la machine du client et taper la commande suivante pour générer une paire de clés publique / privée :

Code : Console

```
ssh-keygen -t rsa
```

Vous pouvez remplacer `rsa` par `dsa` si vous voulez utiliser l'autre algorithme de cryptage, mais ça n'a pas vraiment d'importance ici.

Lorsque vous tapez cette commande, vous allez voir plusieurs messages s'afficher et il vous sera demandé quelques petites précisions :

Code : Console

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mateo21/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mateo21/.ssh/id_rsa.
Your public key has been saved in /home/mateo21/.ssh/id_rsa.pub.
The key fingerprint is:
b7:22:94:aa:8c:fb:d3:ef:53:86:df:b9:37:40:bd:4d mateo21@mateo21-laptop
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|           .     |
|       .   . . E |
|      o.S..   +  |
|     o. o.... .  |
|   .. .+...o     |
| o... ....o o    |
|oo+. oo.   .o .  |
+-----------------+
```

Dans un premier temps, le client génère une paire de clés (« *Generating public/private rsa key pair* »).
Il doit ensuite sauvegarder ces clés dans des fichiers (un pour la clé publique, un pour la clé privée). On vous propose une valeur par défaut : je vous conseille de ne rien changer et de taper simplement `Entrée`.

Ensuite, on vous demande une **passphrase**. C'est une phrase de passe qui va servir à crypter la clé privée pour une meilleure sécurité. Là, vous avez deux choix :

- soit vous tapez `Entrée` directement sans rien écrire, et la clé ne sera pas cryptée sur votre machine ;
- soit vous tapez un mot de passe de votre choix, et la clé sera cryptée.

Tout le monde ne met pas une phrase de passe. En fait, ça dépend du risque que quelqu'un d'autre utilise la machine du client et puisse lire le fichier contenant la très secrète clé privée. Si le PC du client est votre PC chez vous et que personne d'autre ne l'utilise, il y a assez peu de risques (à moins d'avoir un virus, un spyware…). Si c'est en revanche un PC public, je vous recommande vivement de mettre une `passphrase` pour chiffrer la clé qui sera enregistrée.

Si vous hésitez entre les deux méthodes, je vous recommande de rentrer une `passphrase` : c'est quand même la méthode la plus sûre.

Envoyer la clé publique au serveur

Il faut maintenant envoyer au serveur votre clé publique pour qu'il puisse vous crypter des messages.

Votre clé **publique** devrait se trouver dans `~/.ssh/id_rsa.pub` (*pub* comme *public*). correspond à votre home (`/home/mateo21/` dans mon cas). Notez que `.ssh` est un dossier caché.
Votre clé **privée**, elle, se trouve dans `~/.ssh/id_rsa`. Ne la communiquez à personne ! Elle est normalement cryptée si vous avez entré une `passphrase`, ce qui constitue une sécurité de plus.

Vous pouvez déjà vous rendre dans le dossier `.ssh`, pour commencer :

Code : Console

```
cd  /.ssh
```

Si vous faites un `ls`, vous devriez voir ceci :

Code : Console

```
$ ls
id_rsa   id_rsa.pub   known_hosts
```

Les trois fichiers sont :

- `id_rsa` : votre clé privée, qui doit rester secrète. Elle est cryptée si vous avez rentré une `passphrase` ;
- `id_rsa.pub` : la clé publique que vous pouvez communiquer à qui vous voulez, et que vous devez envoyer au serveur ;
- `known_hosts` : c'est la liste des `fingerprint` que votre PC de client tient à jour. Ça lui permet de se souvenir de l'identité des serveurs et de vous avertir si, un jour, votre serveur est remplacé par un autre (qui pourrait être celui d'un pirate !). Je vous en ai déjà parlé un peu plus tôt.

L'opération consiste à envoyer la clé publique (`id_rsa.pub`) au serveur et à l'ajouter à son fichier `authorized_keys` (clés autorisées). Le serveur y garde une liste des clés qu'il autorise à se connecter.

Le plus simple pour cela est d'utiliser la commande spéciale `ssh-copy-id`. Utilisez-la comme ceci :

Code : Console

```
ssh-copy-id -i id_rsa.pub login@ip
```

Remplacez-y votre `login` et l'`ip` de votre serveur.

Code : Console

```
$ ssh-copy-id -i id_rsa.pub mateo21@88.92.107.7
mateo21@88.92.107.7's password:
Now try logging into the machine, with "ssh 'mateo21@localhost'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

Si vous devez vous connecter au serveur par un autre port que celui par défaut, basez-vous sur la commande suivante : `ssh-copy-id -i id_rsa.pub "-p 14521 mateo21@88.92.107.7"`.

On vous demande votre mot de passe (celui de votre compte, pas la `passphrase`). En fait, vous vous connectez par mot de passe encore une fois, pour pouvoir ajouter votre clé publique sur le serveur.

La clé est ensuite automatiquement ajoutée à `~/.ssh/authorized_keys` sur le serveur. On vous invite à vérifier si

l'opération s'est bien déroulée en ouvrant le fichier `authorized_keys`, ce que vous pourrez faire plus tard si vous le voulez.

Se connecter !

Maintenant, connectez-vous au serveur comme vous le faisiez auparavant :

Code : Console

```
ssh login@ip
```

Par exemple :

Code : Console

```
$ ssh mateo21@88.92.107.7
Enter passphrase for key '/home/mateo21/.ssh/id_rsa':
```

On vous demande la phrase de passe pour décrypter votre clé privée. Entrez-la.

Normalement, si tout va bien, vous devriez être alors connectés au serveur.



Ou je suis le dernier des nuls, ou alors c'est ce système qui est nul. Auparavant, on me demandait mon mot de passe. Maintenant, on me demande une phrase de passe pour décrypter la clé privée. Où est le progrès ???

Je comprends votre frustration. ;-)
En fait, si vous n'aviez pas mis de phrase de passe, on ne vous aurait rien demandé et vous auriez été directement connectés. Heureusement, il y a une solution pour ceux qui ont choisi la sécurité en utilisant une phrase de passe, mais qui ne veulent quand même pas avoir à l'entrer à chaque fois : l'agent SSH.

L'agent SSH

L'agent SSH est un programme qui tourne en arrière-plan en mémoire. Il retient les clés privées pendant toute la durée de votre session.

Tout ce que vous avez à faire est de lancer le programme `ssh-add` sur le PC du client :

Code : Console

```
$ ssh-add
Enter passphrase for /home/mateo21/.ssh/id_rsa:
Identity added: /home/mateo21/.ssh/id_rsa (/home/mateo21/.ssh/id_rsa)
```

Il va automatiquement chercher votre clé privée. Pour la décrypter, il vous demande la `passphrase`. Entrez-la.

Maintenant que c'est fait, chaque fois que vous vous connecterez à un serveur, vous n'aurez plus besoin d'entrer la `passphrase`. Essayez de vous connecter à votre serveur pour voir !



L'intérêt de l'agent SSH est qu'il ne vous demande la `passphrase` qu'une seule fois au début. Ensuite, vous pouvez vous connecter plusieurs fois sur le même serveur, ou même sur plusieurs serveurs différents, le tout sans avoir besoin de retaper votre `passphrase` !

Authentification par clé depuis Windows (PuTTY)

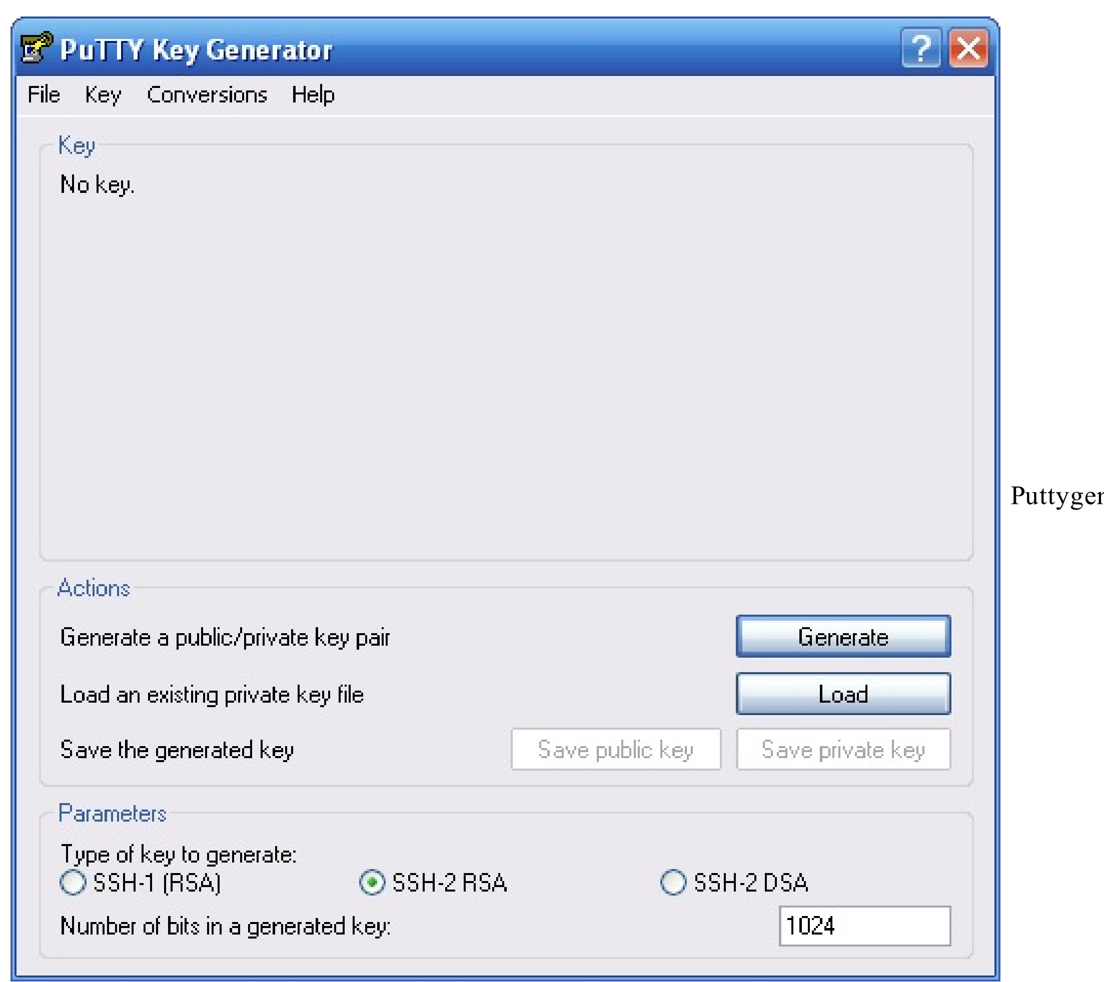
Il est tout à fait possible d'utiliser l'authentification par clé avec PuTTY. C'est là justement qu'il est recommandé d'avoir téléchargé le programme d'installation, et non pas juste le programme principal `putty.exe`.

Le principe est le même que sous Linux : il faut d'abord que l'on génère une paire de clés sur le PC du client, puis qu'on les envoie au serveur. Nous retrouverons aussi un équivalent de l'agent SSH pour éviter d'avoir à entrer une `passphrase` à chaque fois.

Commençons par la génération des clés.

Générer une paire de clés (publique et privée) avec Puttygen

Normalement, vous devriez avoir installé un programme appelé Puttygen (figure suivante – il se trouvait dans le gestionnaire d'installation de PuTTY). Lancez-le.



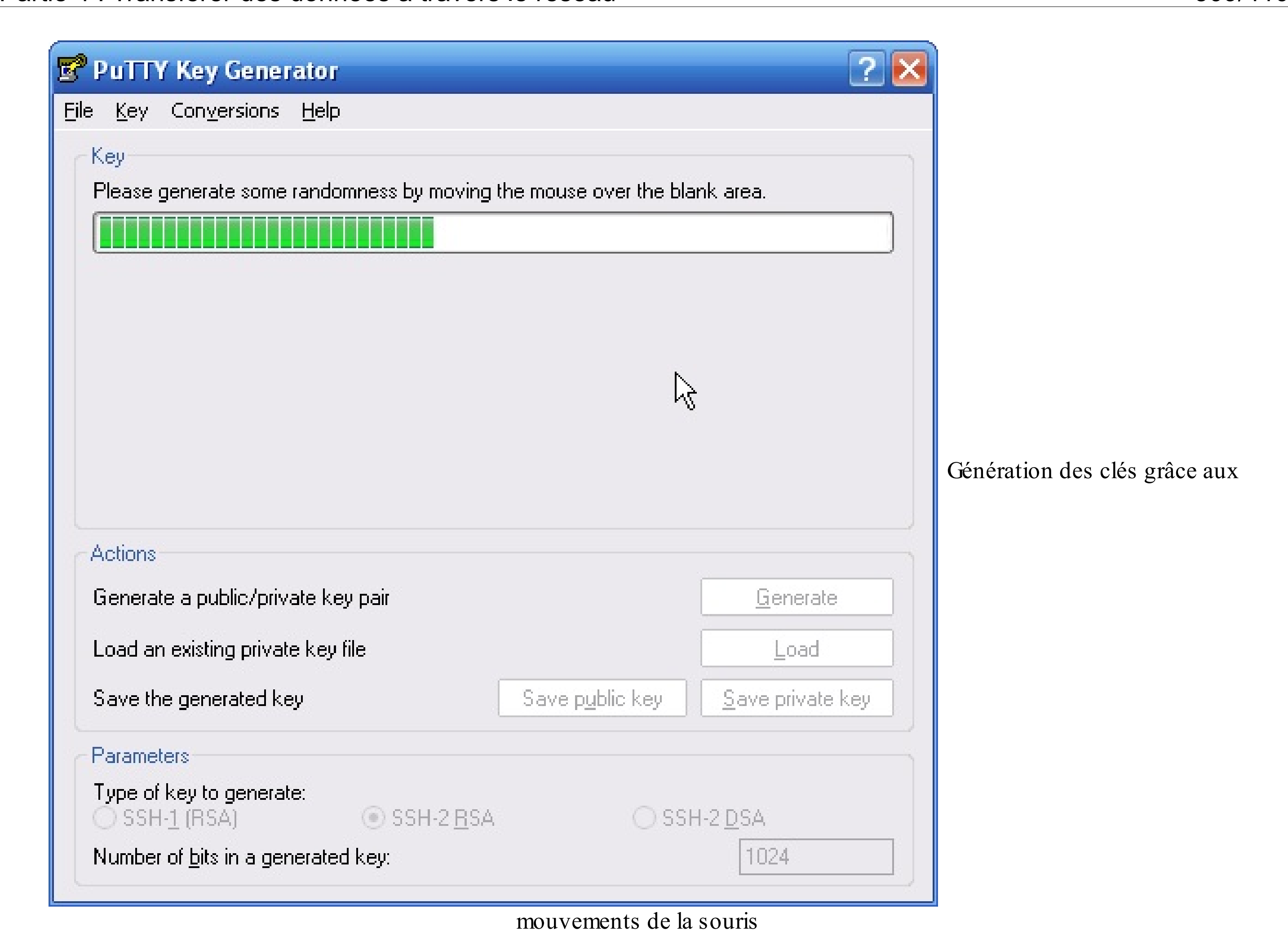
Puttygen permet de générer une paire de clés

En bas de la fenêtre, vous pouvez choisir vos paramètres : algorithme de cryptage et puissance du cryptage. Les valeurs par défaut (ici RSA 1024 bits) sont tout à fait convenables. Vous pouvez les changer, mais sachez qu'elles sont sûres et que vous pouvez vous en contenter.

Cliquez sur le bouton « Generate ». Le programme va générer une paire de clés (publique et privée).

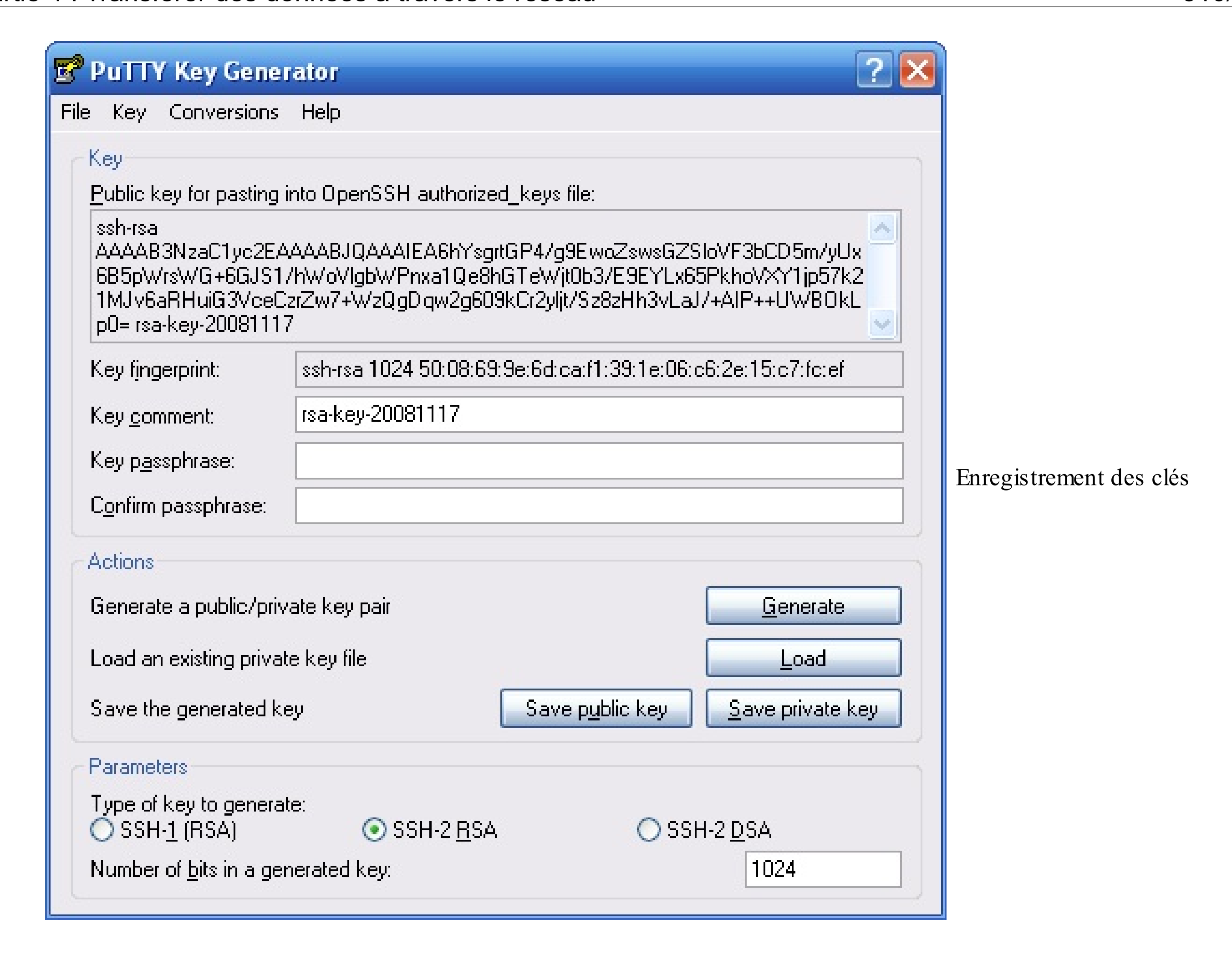


Pour l'aider à générer cette paire, le programme vous propose quelque chose d'assez amusant : vous devez bouger la souris dans la fenêtre (figure suivante)



Génération des clés grâce aux mouvements de la souris

Une fois que c'est fait, on vous affiche la clé publique (figure suivante) :



Enregistrement des clés

Comme vous le voyez, cela ne me dérange pas que tout le monde voie ma clé publique. Le principe, c'est justement que tout le monde peut voir cette clé, mais ne peut rien en faire. Par contre, la clé privée doit rester secrète.

Vous pouvez choisir d'entrer une `passphrase` ou non. Comme je vous l'ai expliqué plus tôt, cela renforce la sécurité en cryptant la clé privée.
Saisissez la `passphrase` dans les champs « Key passphrase » et « Confirm passphrase ».

Ensuite, enregistrez la clé publique dans un fichier en cliquant sur « Save public key ». Vous pouvez nommer ce fichier comme vous voulez, par exemple `cle.pub`. Enregistrez-le où vous voulez.
Puis enregistrez la clé privée en cliquant sur « Save private key ». Donnez-lui l'extension `.ppk` : `cle.ppk` par exemple.

Ne fermez pas encore Puttygen.

Envoyer la clé publique au serveur

Comme sous Linux tout à l'heure, il faut envoyer la clé publique au serveur pour qu'il nous autorise à nous connecter par clé. Le problème, c'est qu'il n'y a pas de commande pour le faire automatiquement depuis Windows. Il va falloir ajouter la clé à la main dans le fichier `authorized_keys`. Heureusement, ce n'est pas très compliqué.

Ouvrez PuTTY et connectez-vous au serveur comme auparavant (en entrant votre mot de passe habituel). Rendez-vous dans `~/.ssh` :

Code : Console

```
cd  /.ssh
```

Si le dossier `.ssh` n'existe pas, pas de panique, créez-le :

Code : Console

```
mkdir .ssh
```

Rajoutez votre clé publique à la fin du fichier `authorized_keys` (s'il n'existe pas, il sera créé). Vous pouvez utiliser la commande suivante :

Code : Console

```
echo "votre_cle" >> authorized_keys
```



Rappel : votre clé publique est affichée dans Puttygen, que vous ne devriez pas avoir fermé. Pour coller la clé dans la console, utilisez la combinaison de touches `Shift + Inser` plutôt que `Ctrl + V`.

Par exemple :

Code : Console

```
echo "ssh-rsa AAAAB3NzaC1yc2E [...] AAAABJQAP++UWBOkLp0= rsa-key-
20081117" >> authorized_keys
```

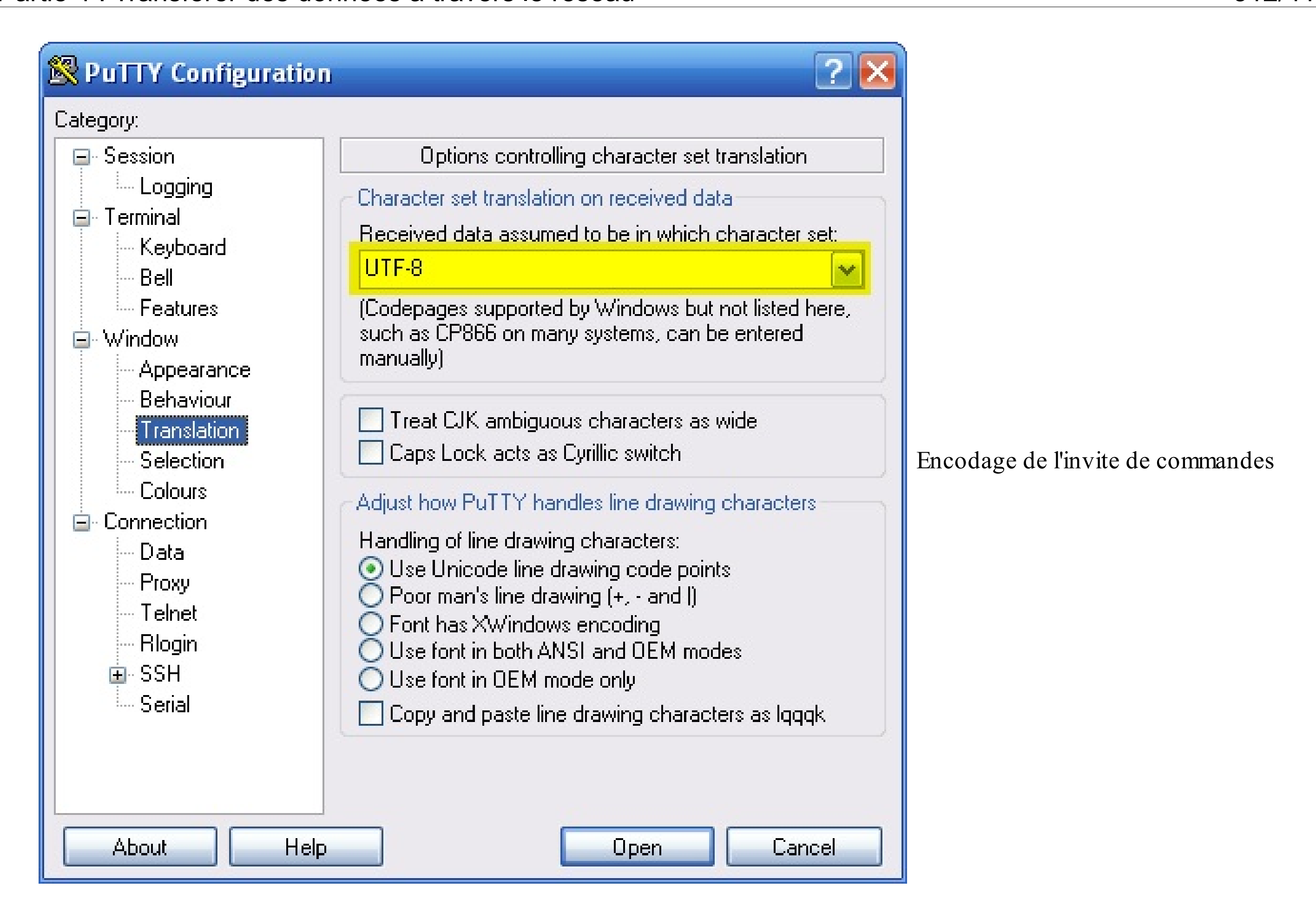
Voilà, c'est fait. ;-)

Déconnectez-vous, et relancez PuTTY. On va maintenant le configurer pour qu'il se connecte à l'aide de la clé.

Configurer PuTTY pour qu'il se connecte avec la clé

Une fois PuTTY ouvert, rendez-vous dans la section `Window → Translation` pour commencer. Ça n'a pas de rapport direct avec les clés, mais cela vous permettra de régler le problème des accents qui s'affichent mal dans la console si vous l'avez rencontré.

Réglez la valeur de la liste déroulante à UTF-8, comme sur figure suivante.



Encodage de l'invite de commandes

La plupart des serveurs encodent désormais les caractères en UTF-8, cela devrait donc vous éviter des soucis d'affichage.

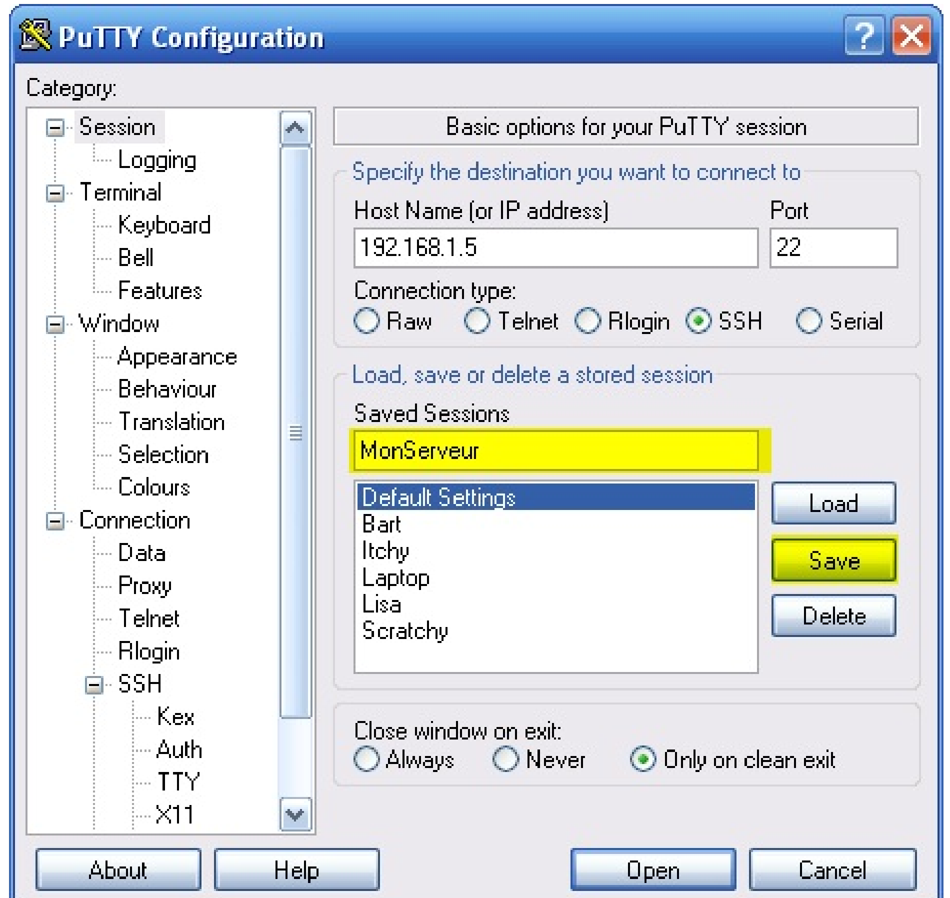
Maintenant, rendez-vous dans `Connection → SSH → Auth`. Cliquez sur le petit bouton « Browse » pour sélectionner votre clé privée (figure suivante).

Sélection de la clé privée

Je vous recommande aussi d'aller dans `Connection` → `Data` et d'entrer votre login dans « Auto-login username », comme la figure suivante vous le montre.

Choix d'un login par défaut

Retournez à l'accueil en cliquant sur la section « Session » tout en haut (figure suivante). Entrez l'IP du serveur. Ensuite, je vous recommande fortement d'enregistrer ces paramètres.

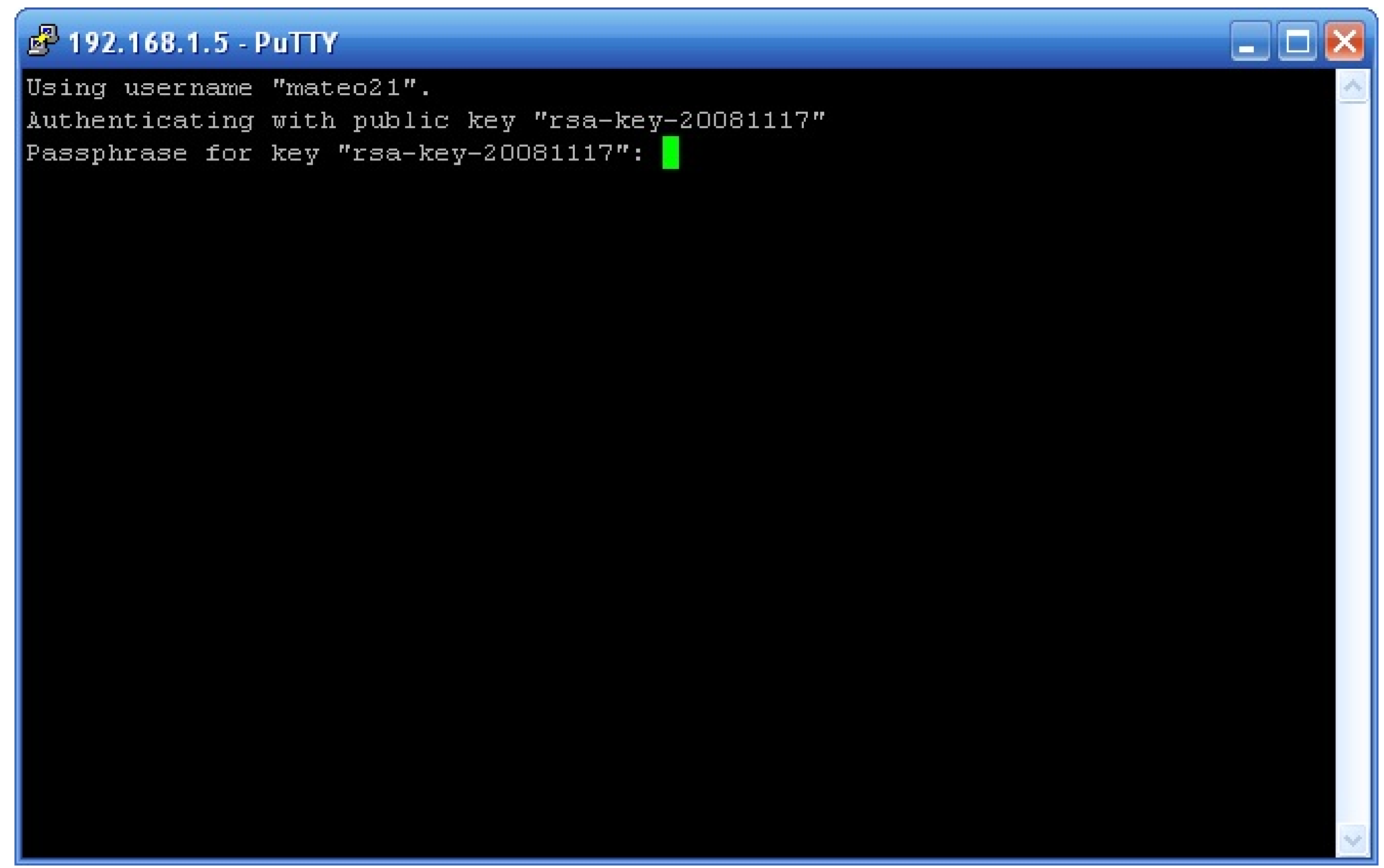


Enregistrement des paramètres dans PuTTY

Donnez un nom à votre serveur (par exemple `MonServeur`) sous « Saved Sessions ». Cliquez ensuite sur « Save ».
À l'avenir, vous n'aurez qu'à double-cliquer sur le nom de votre serveur dans la liste pour vous y connecter directement avec les bons paramètres.

Cliquez sur « Open » pour vous connecter au serveur.

Vous devriez voir PuTTY utiliser automatiquement votre pseudo, puis vous demander votre `passphrase`. Entrez-la pour vérifier que cela fonctionne, comme sur la figure suivante.



PuTTY demande la phrase de passe

Euh… et si je ne veux pas avoir à entrer la `passphrase` à chaque fois ? Non, parce que c'est pareil que d'entrer un mot de passe, là…

En effet, et ma réponse sera la même que pour ceux qui se connectent depuis Linux : il faut utiliser un agent SSH. Ce programme va rester en mémoire et retenir votre clé privée. Il ne vous demandera la `passphrase` qu'une fois au début ; ensuite, vous pourrez vous connecter autant de fois que vous le souhaitez à autant de serveurs que vous voulez sans avoir à entrer quoi que ce soit.

L'agent SSH Pageant

L'agent SSH installé avec PuTTY s'appelle « Pageant ». Je vous recommande de le lancer au démarrage de l'ordinateur automatiquement (il ne prend que 4 Mo en mémoire), en le plaçant dans le dossier `Démarrage` du menu « Démarrer ».

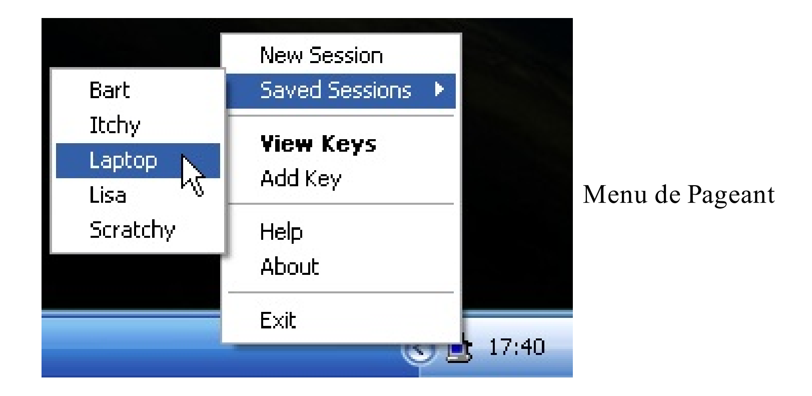
Lorsque vous lancez « Pageant », la petite icône d'un ordinateur avec un chapeau s'ajoute dans la barre des tâches à côté de l'horloge, comme sur la figure suivante.



Pageant

Faites un clic droit dessus, puis cliquez sur « Add key ». On vous demande où se trouve la clé privée (`cle.ppk`). Entrez ensuite la `passphrase`.

C'est bon. Vous avez juste besoin de le faire une fois. Maintenant, vous pouvez vous connecter au serveur que vous voulez en faisant un clic droit sur l'icône, puis en sélectionnant « Saved Sessions » (figure suivante).



Menu de Pageant

On ne vous demandera plus votre clé. :-)



Notez que si l'agent SSH « Pageant » est pratique, il vaut mieux l'arrêter si vous devez vous absenter de votre ordinateur un long moment et que quelqu'un risque de l'utiliser. Sinon, n'importe qui peut se connecter à vos serveurs sans avoir à entrer de mot de passe.
Retenez bien : l'agent SSH est un compromis entre la sécurité et le côté pratique. Il retient les clés pour vous (du moins tant que le programme tourne). Si vous êtes des utilisateurs intensifs de SSH, cela vous fera gagner beaucoup de temps.

Vous pouvez modifier le raccourci qui lance « Pageant » pour que celui-ci charge votre clé privée automatiquement dès son lancement. Faites un clic droit sur l'icône de « Pageant », allez dans « Propriétés ».
Dans le champ « Cible », rajoutez à la fin en paramètre le chemin de la clé à charger. Par exemple :

Code : Console

```
"C:\Program Files\PuTTY\pageant.exe" c:\cle.ppk
```

La clé sera alors chargée dès que vous lancerez « Pageant ».

En résumé

- On peut se connecter à distance à un ordinateur équipé de Linux et accéder à sa console. C'est comme cela que l'on administre les serveurs sous Linux.
- Le PC qui se connecte au serveur équipé de Linux est appelé le client. On peut se connecter à une console Linux à distance depuis n'importe quel autre système d'exploitation (Windows, Mac OS ou Linux).
- Sous Windows, il faut installer le programme PuTTY pour se connecter à distance à un PC équipé de Linux.
- Sous Linux et Mac OS, on utilise la commande `ssh` à laquelle on indique son login et l'adresse IP de la machine. Par exemple : `ssh mateo21@74.141.18.33`.
- Les données qui sont échangées entre le client et le serveur sont cryptées grâce au protocole SSH afin de garantir la confidentialité des échanges.
- Pour éviter de devoir entrer son mot de passe à chaque fois que l'on se connecte au serveur, on peut se créer une paire de clés d'identification. La clé publique ainsi générée doit être envoyée sur le serveur, la clé privée restant sur le PC du client. La connexion se fait alors sans mot de passe et reste sécurisée.

 Transférer des fichiers

Vous avez appris à vous connecter à un serveur à distance avec SSH. Désormais, grâce au réseau, vous pouvez exécuter des commandes sur un ordinateur en prenant le contrôle à distance.

On continue ici notre découverte du monde fabuleux des réseaux sous Linux. Un monde un peu particulier, comme vous avez pu le découvrir : de gros efforts sont faits pour assurer la sécurité des données grâce au cryptage, ce qui permet d'éviter de se faire voler ses données personnelles, comme son mot de passe par exemple.

Dans ce chapitre, nous allons mettre l'accent sur le transfert de fichiers : comment télécharger un fichier ? Comment se connecter à un FTP, lire et télécharger des fichiers ? Et surtout, comment copier des fichiers de manière sécurisée ?

wget : téléchargement de fichiers

Nous commençons par une commande simple à utiliser, du moins en apparence : `wget`. Elle permet de télécharger des fichiers directement depuis la console.

Il suffit d'indiquer l'adresse HTTP ou FTP d'un fichier à télécharger :

Code : Console

```
$ wget http://cdimage.debian.org/debian-cd/4.0_r5/i386/iso-cd/ debian-
40r5-i386-businesscard.iso
```

Une barre de progression du téléchargement devrait alors s'afficher dans la console :

Code : Console

```
$ wget http://cdimage.debian.org/debian-cd/4.0_r5/i386/iso-cd/ debian-
40r5-i386-businesscard.iso
--2008-12-05 12:43:25--  http://cdimage.debian.org/debian-
cd/4.0_r5/ i386/iso-cd/debian-40r5-i386-businesscard.iso
Résolution de cdimage.debian.org... 130.239.18.173, 130.239.18.137
Connexion vers cdimage.debian.org|130.239.18.173|:80... connecté.
requête HTTP transmise, en attente de la réponse... 302 Found
Emplacement: http://saimei.acc.umu.se/debian-cd/4.0_r5/i386/iso-
cd/ debian-40r5-i386-businesscard.iso [suivant]
--2008-12-05 12:43:25--  http://saimei.acc.umu.se/debian-
cd/4.0_r5/ i386/iso-cd/debian-40r5-i386-businesscard.iso
Résolution de saimei.acc.umu.se... 130.239.18.138
Connexion vers saimei.acc.umu.se|130.239.18.138|:80... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Longueur: 34181120 (33M) [application/octet-stream]
Saving to: `debian-40r5-i386-businesscard.iso'

38% [==============>                        ] 13 208 331   117K/s  eta 70s
```

Les informations au début sont assez nombreuses et ne nous intéressent pas vraiment. Elles indiquent simplement comment le programme a communiqué avec le serveur qui possédait le fichier.

En bas, vous avez, dans l'ordre et sur la dernière ligne :

1. une barre de progression qui se met à jour ;
2. le nombre d'octets téléchargés ;
3. la vitesse de téléchargement ;
4. le temps restant estimé (`eta`).

Vous pouvez arrêter le téléchargement à tout moment en utilisant la combinaison `Ctrl + C` que vous avez déjà découverte.

 Comment puis-je récupérer l'adresse du fichier à télécharger pour la donner à `wget` ?

Le plus simple est d'ouvrir un navigateur web tel que Firefox là où vous avez accès à un environnement graphique et de faire un clic droit sur le lien du fichier que vous voulez télécharger, pour enfin sélectionner « Copier l'adresse du lien ». Vous pouvez ensuite le coller dans la console.

Notez qu'il existe aussi des navigateurs en console tels que `lynx` (plutôt basique) et `links` (assez complet) que vous pouvez télécharger et essayer si vous le souhaitez.

Reprendre un téléchargement arrêté

Si vous voulez reprendre un téléchargement arrêté, utilisez l'option `-c` :

Code : Console

```
$ wget -c http://cdimage.debian.org/debian-cd/4.0_r5/i386/iso-cd/ debian-
40r5-i386-businesscard.iso
```

Pour que cela fonctionne, il ne faut bien évidemment pas supprimer le bout de fichier téléchargé sur votre disque. ;-)
Si la reprise a fonctionné vous devriez voir une barre de progression comme celle-ci :

Code : Console

```
71% [++++++++++++++===========>              ] 24 450 216   470K/s   eta 88s
```

Les +++ correspondent à la partie précédemment téléchargée. Cela vous confirme que la reprise a bien fonctionné.

Lancer un téléchargement en tâche de fond

Enfin, si vous voulez que le téléchargement soit envoyé en tâche de fond dès le début, il y a la technique du `nohup` que l'on connaît et qui s'applique à toutes les commandes, mais vous pouvez aussi utiliser l'option `--background` :

Code : Console

```
$ wget --background -c http://cdimage.debian.org/debian-
cd/4.0_r5/ i386/iso-cd/debian-40r5-i386-businesscard.iso
Poursuite à l'arrière plan, pid 8422.
La sortie sera écrite vers « wget-log ».
```

Comme indiqué, l'avancement du téléchargement sera écrit dans un fichier `wget-log`.



`wget` propose une quantité impressionnante d'options et je ne peux donc pas toutes les traiter. Sachez que vous pouvez notamment l'utiliser pour télécharger des pages web. Consultez le manuel pour en savoir plus car il y a de quoi faire : `man wget`.
Un des avantages de `wget` est que vous avez toujours une barre de progression, et cela même si vous téléchargez un fichier depuis un serveur FTP. Ça paraît bête, mais, contrairement à `wget`, le programme `ftp` que l'on verra plus loin ne donne pas l'avancement du téléchargement !

scp : copier des fichiers sur le réseau

Vous connaissez la commande `cp` ? Elle permet de copier des fichiers sur votre disque dur.
Eh bien voici `scp` (*Secure CoPy*), qui permet de copier des fichiers d'un ordinateur à un autre à travers le réseau ! Le tout de manière sécurisée, bien sûr. :-)



Il existe aussi `rcp` (*Remote CoPy*) qui fait la même chose mais sans aucun cryptage. Son utilisation est déconseillée.



scp s'utilise quasiment comme ssh. D'ailleurs ce n'est pas un hasard car scp s'appuie sur ssh pour fonctionner. Là où ssh sert à ouvrir une console à distance (un shell), scp est spécialement conçue pour copier des fichiers d'un ordinateur à un autre.

On l'utilise comme ceci :

Code : Console

```
scp fichier_origine copie_destination
```

Le premier élément à indiquer est la position du fichier que l'on veut copier. Le second élément correspond au répertoire de destination où il doit être copié.

Chacun de ces éléments peut s'écrire sous la forme suivante : login@ip:nom_fichier. Le login et l'IP sont facultatifs. Si vous n'écrivez ni login ni IP, scp considérera que le fichier se trouve sur votre ordinateur.



Vous pouvez remplacer l'IP par un nom d'hôte (un nom de domaine) si vous en avez un qui est plus facile à retenir. Par exemple, le serveur Lisa du Site du Zéro peut être appelé en écrivant lisa.simple-it.fr au lieu d'une IP compliquée comme 85.123.10.201.

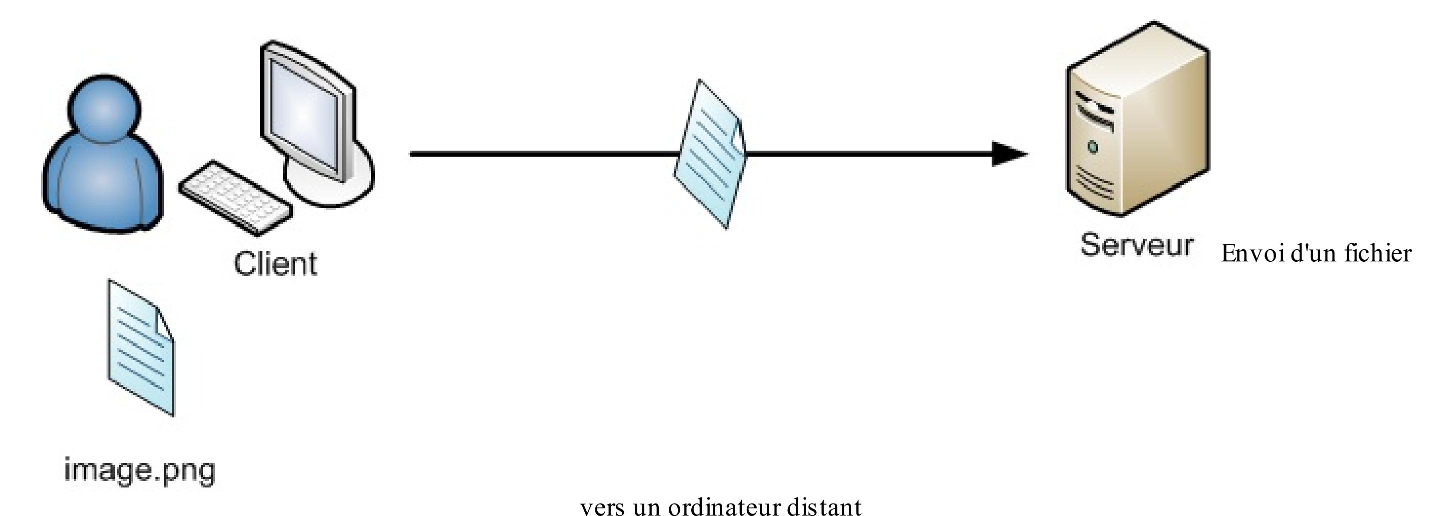
Copier un fichier de votre ordinateur vers un autre

Prenons un cas concret (figure suivante) pour que vous puissiez mieux voir comment ça s'utilise :

Code : Console

```
scp image.png mateo21@85.123.10.201:/home/mateo21/images/
```

Ici, je demande à copier le fichier image.png qui se trouve sur mon ordinateur vers un autre ordinateur dont l'IP est 85.123.10.201. Sur cet autre ordinateur, le fichier sera placé dans le dossier /home/mateo21/images/.



Envoi d'un fichier vers un ordinateur distant

Notez que l'on peut utiliser le symbole ~ pour indiquer « mon répertoire personnel » (/home/mateo21/). D'autre part, si cet autre ordinateur a un nom d'hôte facile à retenir, j'aurais tendance à l'utiliser à la place de l'IP. J'aurais donc pu écrire quelque chose comme ce qui suit et qui aurait été identique :

Code : Console

```
scp image.png mateo21@lisa.simple-it.fr:~/images/
```

Lorsque vous lancez la commande, `scp` essaiera de se connecter au serveur ayant l'IP indiquée avec le login que vous avez demandé (`mateo21`, dans mon cas). On vous demandera alors votre mot de passe ou, mieux, `scp` utilisera votre clé privée si elle existe.
Nous avons vu que, combiné à l'agent SSH, cela nous évitait d'avoir à retaper notre `passphrase` à chaque fois !

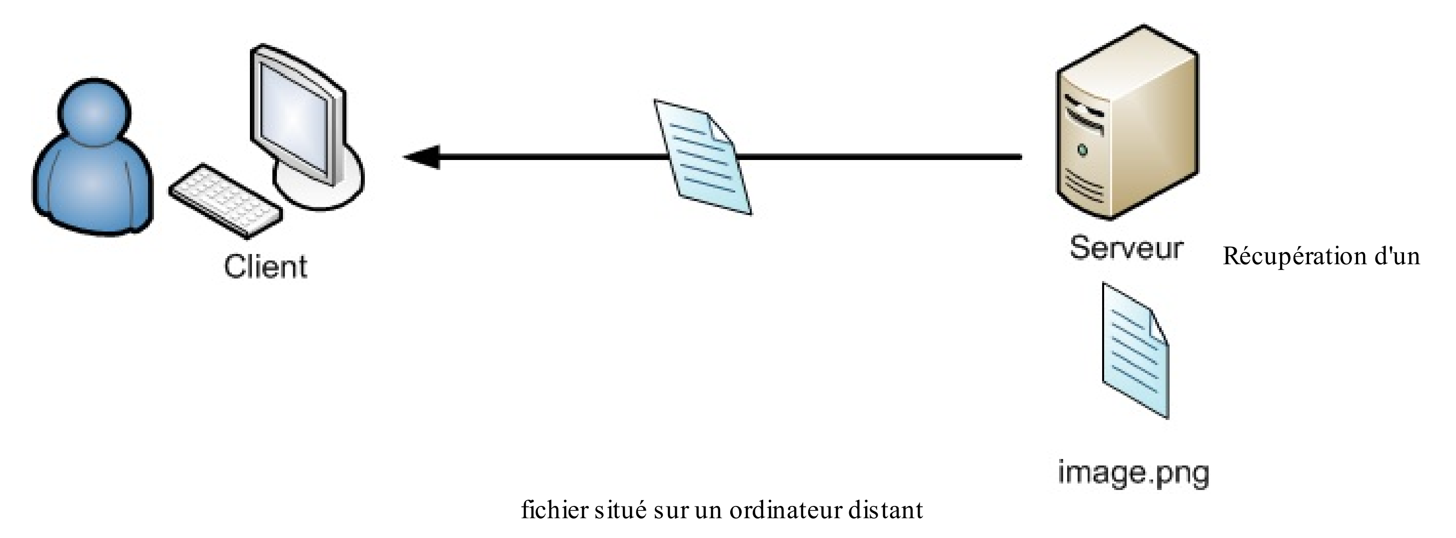
Copier un fichier d'un autre ordinateur vers le vôtre

On peut également faire le contraire (figure suivante), c'est-à-dire récupérer un fichier qui se trouve sur un autre ordinateur et le placer chez nous :

Code : Console

```
scp mateo21@85.123.10.201:image.png copie_image_sur_mon_pc.png
```

Dans cet exemple, je copie le fichier `image.png` qui se trouve sur le serveur dont l'IP est `85.123.10.201` et place cette copie sur mon propre ordinateur sous le nom `copie_image_sur_mon_pc.png`.



Récupération d'un fichier situé sur un ordinateur distant

Si je veux, je peux aussi copier le fichier sans en changer le nom :

Code : Console

```
scp mateo21@85.123.10.201:image.png .
```

Notez le point à la fin. Il signifie « copier dans le répertoire dans lequel je me trouve ». Le fichier `image.png` sera donc placé sur mon ordinateur dans le dossier actuel.

Le piège du port

Si le serveur SSH auquel vous essayez de vous connecter n'est pas sur le port standard (22), il faudra indiquer le numéro du port avec l'option `-P` :

Code : Console

```
scp -P 16296 mateo21@85.123.10.201:image.png .
```



La commande ssh a aussi une option pour indiquer un port ; vous vous en souvenez peut-être, c'est… -p ! Faites donc attention car avec ssh c'est un « p » minuscule qu'il faut utiliser alors qu'avec scp c'est un « P » majuscule ! Je peux vous dire que je me suis trompé un bon nombre de fois.

ftp & sftp : transférer des fichiers

Le FTP (*File Transfer Protocol*) est un protocole permettant d'échanger des fichiers sur le réseau. Il est assez ancien (1985) et toujours utilisé à l'heure actuelle pour transférer des fichiers.

On l'utilise en général dans deux cas.

- Pour télécharger un fichier depuis un serveur FTP public. En général, les navigateurs web font cela de manière automatique et transparente lorsque vous cliquez sur un lien de téléchargement.
  La connexion se fait alors en **mode anonyme**.
- Pour transférer des fichiers vers un serveur FTP privé (et éventuellement en télécharger aussi). Lorsque l'on prend un hébergement pour son site web, l'hébergeur nous donne en général des accès FTP pour aller y déposer les fichiers du site.
  La connexion se fait donc en **mode authentifié**.

Tout le monde n'a pas forcément accès à un serveur FTP privé, aussi je vous propose pour les exemples suivants de vous connecter à un serveur FTP public (rassurez-vous, si vous voulez vous connecter à un FTP privé, la méthode reste la même).



Nous nous intéressons ici au fonctionnement du FTP en ligne de commande. Bien sûr, il existe des logiciels graphiques qui font la même chose, comme par exemple FileZilla.

Connexion à un serveur FTP

Essayons de nous connecter au serveur FTP de Debian, accessible à l'adresse suivante : ftp://ftp.debian.org.

Code : Console

```
$ ftp ftp.debian.org
```

Le serveur FTP devrait répondre en vous demandant un login et un mot de passe. Pour les serveurs FTP publics, le login à utiliser est toujours anonymous (anonyme).

Code : Console

```
$ ftp ftp.debian.org
Connected to ftp.debian.org.
220 saens.debian.org FTP server (vsftpd)
Name (ftp.debian.org:mateo21): anonymous
331 Please specify the password.
Password:
```

Pour le mot de passe, peu importe ce que vous mettez, vous serez acceptés. :-)

Vous devriez alors voir s'afficher un message de bienvenue se terminant par :

Code : Console

```
230 Login successful.
Remote system type is UNIX.
```

```
Using binary mode to transfer files.
ftp>
```

Vous avez maintenant un **prompt** (il s'agit du terme anglais pour *invite de commandes*) `ftp>` qui vous permet de rentrer des commandes FTP.

Se déplacer au sein du serveur FTP

Vous savez quoi ? Bonne nouvelle : les commandes que vous pouvez utiliser sont pour la plupart les mêmes que celles que vous connaissez. :-)

Citons dans le lot :

- `ls` : affiche le contenu du répertoire actuel ;
- `pwd` : affiche le chemin du répertoire actuel ;
- `cd` : change de répertoire.

Avec ces commandes, vous devriez déjà pouvoir vous balader sur le serveur FTP.
Faites un `ls` pour voir :

Code : Console

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxrwsr-x    7 1176     1176         4096 Dec 05 09:10 debian
226 Directory send OK.
```

Les lignes commençant par un numéro sont des messages envoyés par le serveur FTP. Vous noterez que les fichiers s'affichent comme si l'on avait écrit `ls -l`.

Il y a seulement un répertoire, rendez-vous donc dans `debian` :

Code : Console

```
ftp> cd debian
250 Directory successfully changed.
```

Affichez à nouveau le contenu :

Code : Console

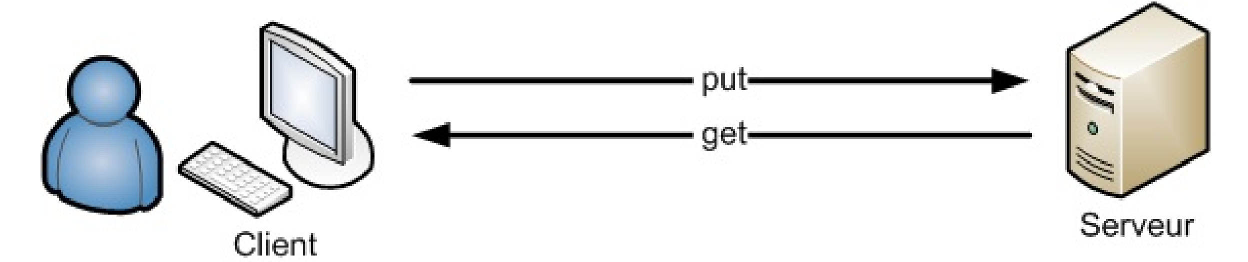
```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--    1 1176     1176          940 Oct 27 20:29 README
-rw-rw-r--    1 1176     1176         1290 Dec 04  2000 README.CD-
manufacture
-rw-rw-r--    1 1176     1176         2426 Oct 27 20:29 README.html
-rw-r--r--
    1 1176     1176       124286 Dec 03 19:52 README.mirrors.html
-rw-r--r--
    1 1176     1176        62059 Dec 03 19:52 README.mirrors.txt
drwxr-sr-x    9 1176     1176         4096 Nov 16 18:56 dists
drwxr-sr-x    3 1176     1176         4096 Nov 11 22:16 doc
drwxr-sr-x    3 1176     1176         4096 Dec 05 09:08 indices
-rw-rw-r--    1 1176     1176      4557196 Dec 05 08:49 ls-lR.gz
```

```
-rw-r--r--    1 1176     1176       154934 Dec 05 08:49 ls-lR.patch.gz
drwxr-sr-x    5 1176     1176         4096 Nov 11 22:16 pool
drwxr-sr-x    4 1176     1176         4096 Nov 18 09:04 project
226 Directory send OK.
```

Le transfert de fichiers

Si vous souhaitez récupérer un fichier ou en envoyer un, il y a deux commandes à connaître (figure suivante) :

- `put` : envoie un fichier vers le serveur ;
- `get` : télécharge un fichier depuis le serveur.





Notez qu'il est impossible d'utiliser `put` sur les serveurs FTP publics comme celui auquel nous sommes connectés. Seul le téléchargement de fichiers est autorisé.
D'autres commandes, comme celle qui permet de changer les `chmod` des fichiers, ne sont pas activées non plus.

Vous pouvez par exemple récupérer le fichier `README` en écrivant `get README` :

Code : Console

```
ftp> get README
local: README remote: README
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for README (940 bytes).
226 File send OK.
940 bytes received in 0.00 secs (918.9 kB/s)
```

Et voilà ! Le fichier se trouve maintenant sur votre ordinateur.

Il a été téléchargé dans le dossier dans lequel vous vous trouviez sur votre ordinateur. Pour savoir dans quel dossier vous êtes **chez vous**, tapez `!pwd` :

Code : Console

```
ftp> !pwd
/home/mateo21
```

Si vous voulez changer de dossier **chez vous**, utilisez `!cd`. Pour lister les fichiers **chez vous**, utilisez `!ls`. Bref, vous m'avez compris, il suffit de faire précéder les commandes d'un point d'exclamation pour qu'elles s'exécutent chez vous et non sur le serveur FTP.

Les autres commandes

Il existe de nombreuses autres commandes FTP, nous n'allons pas toutes les voir.

Tapez `man ftp` pour obtenir un aperçu des commandes disponibles. Vous noterez que toutes ne sont pas identiques à celles que vous connaissez. Par exemple, pour supprimer un fichier, ce n'est pas `rm` mais… `delete` ! D'autres vous seront familières : `mkdir` permet de créer un dossier, par exemple.

Pour quitter le serveur et vous déconnecter, vous avez le choix entre la bonne vieille combinaison de touches `Ctrl + D` qui commande la fermeture de la session ou encore les commandes `bye`, `exit` et `quit`, qui sont analogues.

`sftp` : un FTP sécurisé

Le protocole FTP a un défaut : il n'est pas sécurisé ; les données ne sont pas cryptées. Quelqu'un ayant accès au réseau pourrait alors intercepter le contenu des fichiers que vous échangez ou encore votre mot de passe lors de la connexion.

Pour remédier à cela, on a inventé `sftp`, qui repose sur SSH pour sécuriser la connexion :

Code : Console

```
sftp login@ip
```

Par exemple :

Code : Console

```
sftp mateo21@lisa.simple-it.fr
```

On vous demandera alors votre mot de passe (bien entendu, la clé publique sera utilisée, si elle est présente).

Une fois que vous serez connectés, les commandes sont presque les mêmes que pour le FTP. Vous retrouverez notamment `get` et `put` pour échanger des fichiers. Sachez que les commandes sont globalement plus puissantes et pratiques en SFTP qu'en FTP.
Méfiez-vous toutefois, certaines commandes changent ! Par exemple, pour supprimer un fichier, ce n'est plus `delete` mais à nouveau `rm` ! Lisez le manuel pour plus d'informations : `man sftp`.



Pour se connecter en SFTP, on utilise le même port que SSH (soit 22 par défaut). Si votre serveur SSH fonctionne sur un autre port, vous devrez le préciser comme ceci : `sftp -oPort=27401 mateo21@serveur`.

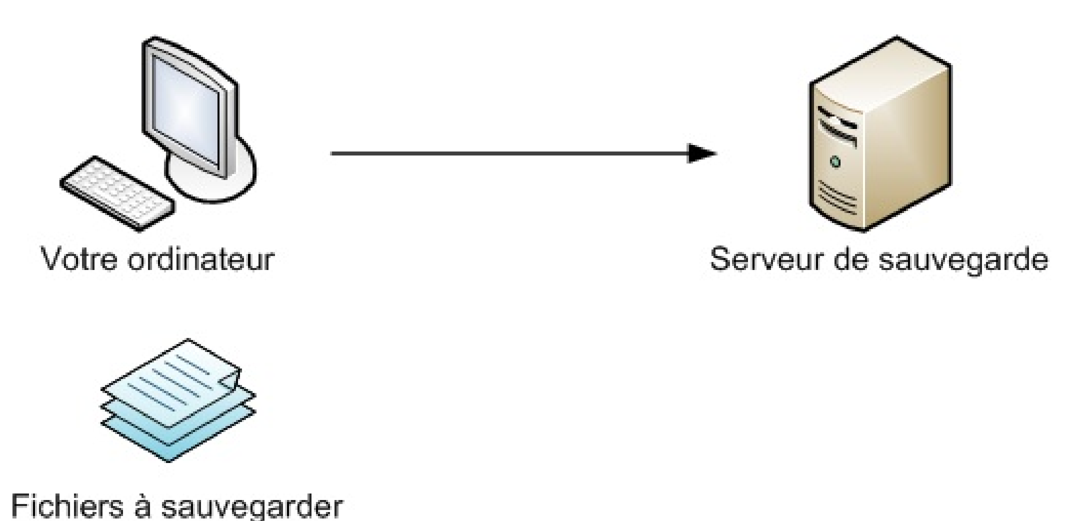
À l'heure actuelle, le SFTP reste assez peu utilisé. Les hébergeurs web utilisent toujours le FTP classique alors que la plupart des logiciels graphiques comme FileZilla sont pourtant capables de se connecter en SFTP.

rsync : synchroniser des fichiers pour une sauvegarde

`rsync` est un programme assez simple à utiliser et pourtant très puissant. Il permet d'effectuer une synchronisation entre deux répertoires, que ce soit sur le même PC ou entre deux ordinateurs reliés en réseau.

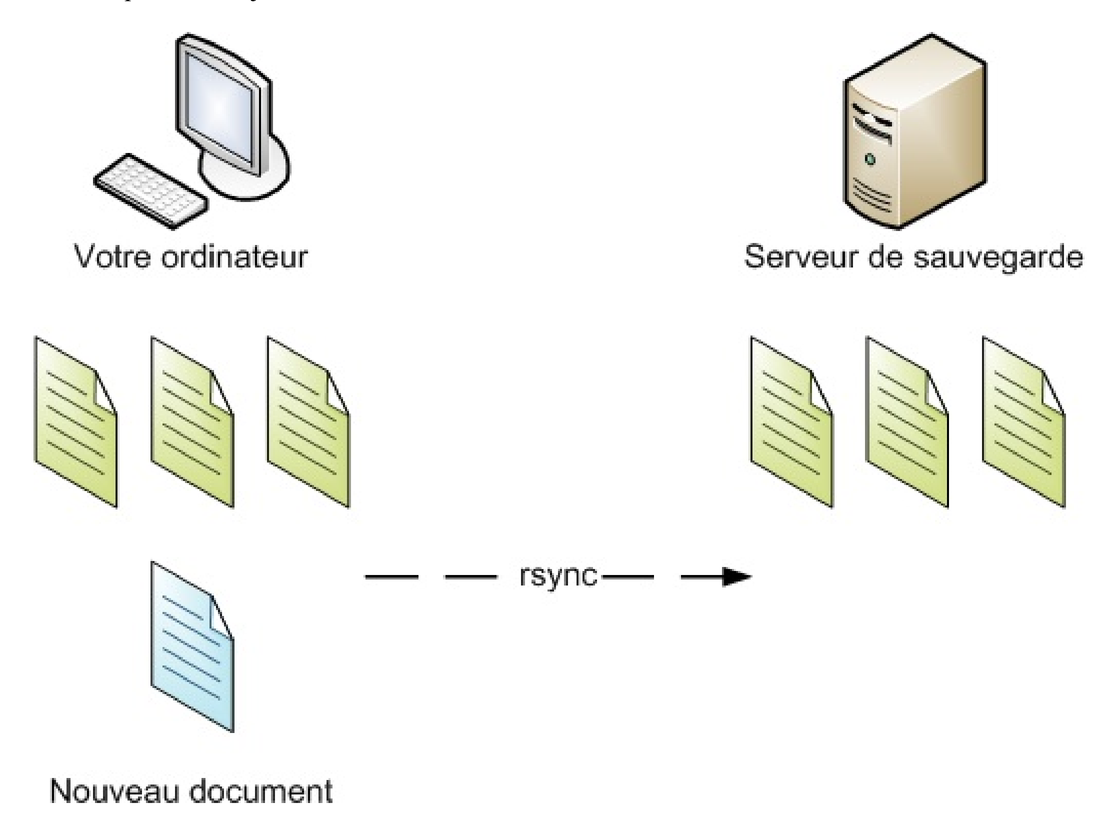
`rsync` est le plus souvent utilisé pour effectuer des **sauvegardes incrémentielles**. Je m'explique.

Par exemple, imaginez que vous souhaitiez sauvegarder régulièrement les fichiers de votre home (répertoire personnel). Ainsi, si un jour un accident survient (perte du PC, par exemple), vous aurez toujours vos documents sauvegardés au chaud sur un serveur quelque part (figure suivante).



D'accord, vous copiez tout votre home sur ce serveur.
Oui mais voilà, votre home contient peut-être 10 Go de fichiers : entre vos vidéos, la musique, vos rapports au format texte et vos photos de vacances, ça va vite.

Sauvegarder ces 10 Go une fois, d'accord. Mais la seconde fois, plutôt que de tout renvoyer, vous souhaiterez peut-être envoyer uniquement les fichiers qui ont été ajoutés ou modifiés, non ?



C'est là que `rsync` intervient. C'est une sorte de `scp` intelligent : il compare et analyse les différences entre deux dossiers puis copie uniquement les changements. C'est ce que veut dire le mot « incrémentiel ».

`rsync` peut être utilisé pour effectuer une sauvegarde entre deux dossiers sur le même ordinateur ou bien entre deux dossiers sur deux ordinateurs différents (figure suivante). En général, on l'utilise plutôt pour sauvegarder entre deux ordinateurs différents, bien sûr.

Dans un premier temps, pour faire simple, nous allons voir comment fonctionne la sauvegarde entre deux dossiers de votre ordinateur puis nous effectuerons ensuite la sauvegarde sur un autre PC.

Sauvegarder dans un autre dossier du même ordinateur

Dans les exemples qui vont suivre, je vais supposer que vous souhaitiez sauvegarder le dossier `Images` dans un dossier `backups`.

Dans le dossier `Images`, il y a quelques photos de vacances :

Code : Console

```
$ ls
espagne1.jpg  italie1.jpg  italie2.jpg  italie3.jpg
```

Vous pouvez créer comme moi des fichiers de test à l'aide de la commande `touch`.

Maintenant, lancez un `rsync` comme ceci :

Code : Console

```
$ rsync -arv Images/ backups/
sending incremental file list
created directory backups
./
espagne1.jpg
italie1.jpg
italie2.jpg
italie3.jpg

sent 268 bytes  received 91 bytes  718.00 bytes/sec
total size is 0  speedup is 0.00
```

Quelques explications concernant les paramètres :

- `-a` : conserve toutes les informations sur les fichiers, comme les droits (`chmod`), la date de modification, etc. ;
- `-r` : sauvegarde aussi tous les sous-dossiers qui se trouvent dans le dossier à sauvegarder ;
- `-v` : mode verbeux, affiche des informations détaillées sur la copie en cours.

Viennent ensuite le nom du dossier à sauvegarder et le répertoire de sauvegarde.

`rsync` analyse le contenu du répertoire de sauvegarde dans un premier temps. Comme celui-ci est vide, vous pouvez constater qu'il y a copié tous les fichiers.

Maintenant, lancez la même commande une seconde fois :

Code : Console

```
$ rsync -arv Images/ backups/
sending incremental file list

sent 109 bytes  received 12 bytes  242.00 bytes/sec
total size is 0  speedup is 0.00
```

Comme vous pouvez le voir, cette fois aucun fichier n'a été envoyé ! En effet, `rsync` étant intelligent, il a détecté qu'il n'y avait aucun changement et donc qu'il n'y avait pas lieu de copier quoi que ce soit.

Testons un peu ce qui se passe si l'on ajoute un fichier :

Code : Console

```
$ touch Images/espagne2.jpg
```

```
$ rsync -arv Images/ backups/
sending incremental file list
./
espagne2.jpg

sent 172 bytes  received 34 bytes  412.00 bytes/sec
total size is 0  speedup is 0.00
```

Le nouveau fichier `espagne2.jpg` a bien été copié ! ;-)
Vous pouvez aussi essayer de modifier un fichier, vous verrez que `rsync` copie bien ceux qui ont été modifiés.

Supprimer les fichiers



J'ai essayé de supprimer un fichier mais celui-ci n'a pas été supprimé dans le répertoire de sauvegarde. Comment faire ?

Par défaut, `rsync` ne supprime pas les fichiers dans le répertoire de copie. Si vous voulez lui demander de le faire, pour que le contenu soit strictement identique, rajoutez `--delete`.
Par exemple, si je supprime le fichier `italie3.jpg` :

Code : Console

```
$ rm Images/italie3.jpg
$ rsync -arv --delete Images/ backups/
sending incremental file list
deleting italie3.jpg

sent 120 bytes  received 12 bytes  264.00 bytes/sec
total size is 4  speedup is 0.03
```

… `rsync` me supprime mon fichier `italie3.jpg` !

Sauvegarder les fichiers supprimés

Peut-être que la suppression du fichier était accidentelle. Si même votre `rsync` supprime le fichier dans le répertoire de sauvegarde, vous n'en aurez plus aucune trace !

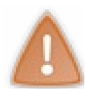
Heureusement, il est possible de garder de côté les fichiers que l'on a supprimés, sait-on jamais, au cas où…

Pour cela, rajoutez l'option `--backup`. Les fichiers supprimés prendront un suffixe dans le répertoire de sauvegarde.
Vous pouvez aussi, pour éviter que ça ne fasse désordre, déplacer les fichiers supprimés dans un dossier qui leur est dédié. Rajoutez `--backup-dir=/chemin/vers/le/repertoire`.

Par exemple :

Code : Console

```
$ rsync -arv --delete --backup --backup-
dir=/home/mateo21/backups _supprimes Images/ backups/
```



Je vous recommande d'indiquer le répertoire `backup-dir` en absolu comme je l'ai fait. Sinon, le répertoire des fichiers supprimés sera placé à l'intérieur du répertoire de sauvegarde et vous risquez d'avoir plus de problèmes qu'autre chose lors de la synchronisation.

`rsync` peut faire bien d'autres choses, comme exclure un dossier de la sauvegarde (option `--exclude`). Je vous laisse lire le manuel pour savoir un peu tout ce que vous pouvez faire.

Sauvegarder sur un autre ordinateur

Intéressons-nous maintenant à la sauvegarde sur un autre ordinateur, parce que là c'est bien joli mais on se sentirait plus en sécurité si les fichiers étaient envoyés ailleurs, sur un autre ordinateur.

L'avantage de `rsync` est qu'il peut copier les fichiers en employant plusieurs méthodes différentes. La plus couramment utilisée, que nous allons choisir ici, est de passer par SSH. Comme quoi, vous le voyez, SSH sert à sécuriser tous types de transferts.

Code : Console

```
$ rsync -arv --delete --backup --backup-
dir=/home/mateo21/fichiers_supprimes Images/ mateo21@IP_du_serveur:mes_backups/
```

Si votre serveur SSH écoute sur un autre port que celui par défaut, il faudra rajouter `-e "ssh -p port"` :

Code : Console

```
$ rsync -arv --delete --backup --backup-
dir=/home/mateo21/fichiers_supprimes Images/ mateo21@IP_du_serveur:mes_backups/ -
e "ssh -p 12473"
```

En résumé

- `wget` permet de télécharger un fichier.
- Pour copier des fichiers d'un ordinateur à un autre, on utilise `scp`. Il fonctionne à l'aide de SSH, donc le transfert est sécurisé.
- On peut se connecter à un serveur FTP avec la commande `ftp` pour y télécharger et y envoyer des fichiers.
- Il existe une alternative sécurisée à FTP qui crypte les échanges grâce à SSH : `sftp`.
- `rsync` permet de synchroniser le contenu de deux dossiers sur un même ordinateur ou sur deux ordinateurs différents. Il est particulièrement utile pour effectuer des sauvegardes.



Analyser le réseau et filtrer le trafic avec un pare-feu

Cechapitre vous propose d'apprendre à maîtriser le trafic réseau qui passe par votre ordinateur. En effet, lorsque vous êtes connectés à l'internet, vous avez régulièrement des applications qui vont se connecter puis télécharger et envoyer des informations. Comment surveiller ce qui se passe ? Quelle application est en train de communiquer et sur quel port ?

Savoir paramétrer un pare-feu est essentiel, que ce soit sur votre PC à la maison ou, à plus forte raison, sur un serveur. Cela vous protège de manière efficace contre les programmes qui voudraient échanger des informations sur le réseau sans votre accord. C'est une mesure de sécurité essentielle qu'il faut connaître et dont aucun administrateur système sérieux ne peut se passer. ;)

Je vous propose de découvrir d'abord quelques outils de base qui vont vous permettre de bien comprendre comment une IP est associée à un nom d'hôte. Puis nous analyserons le trafic en cours avec un outil comme `netstat`. Enfin — et ce ne sera pas le plus facile, je vous préviens — nous apprivoiserons le célèbre pare-feu utilisé sous Linux : `iptables`. Il est assez complexe à paramétrer, mais heureusement des programmes supplémentaires peuvent nous simplifier le travail.

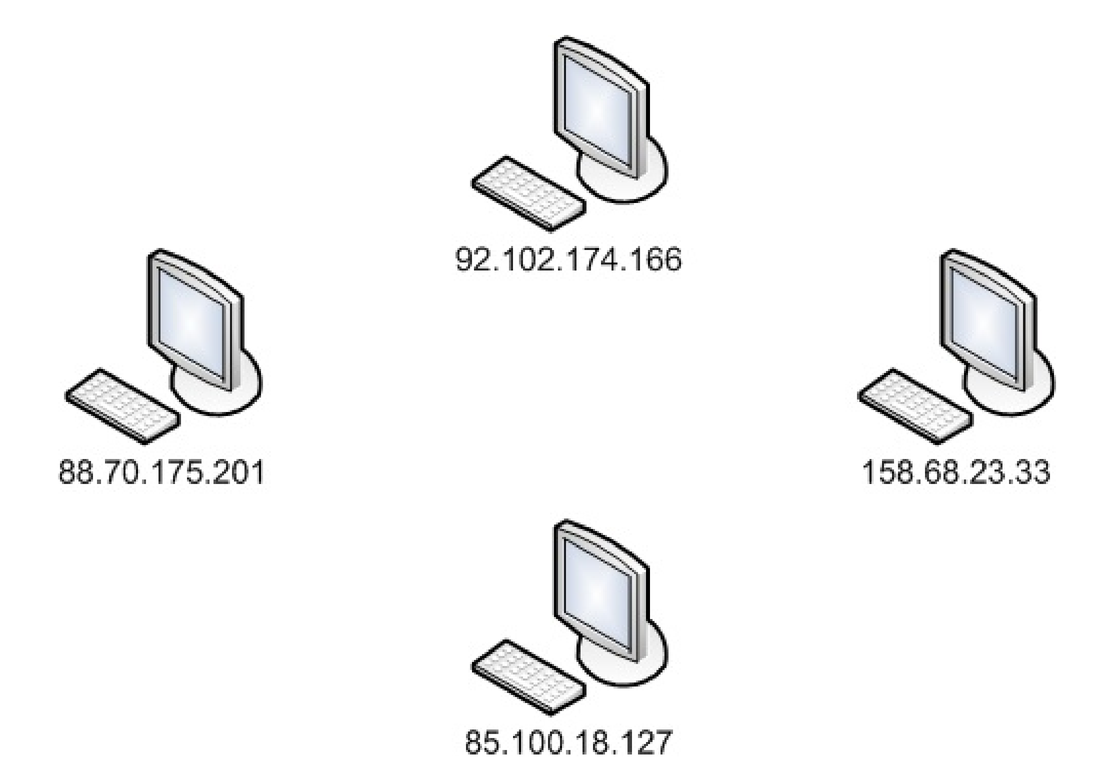
host & whois : qui êtes-vous ?

Comme vous le savez sûrement, chaque ordinateur relié à l'internet est identifié par une adresse IP (figure suivante).

Une adresse IP est une suite de quatre nombres séparés par des points. Par exemple : `86.172.120.28`.



Cette adresse est au format **IPv4**. À l'heure actuelle, c'est encore le type d'IP le plus utilisé, mais ces adresses sont appelées petit à petit à être remplacées par la norme **IPv6**. Bientôt, tout le monde aura donc une IP qui ressemblera plutôt à quelque chose comme ceci : `fe80::209:62fa:fb80:29f2`.



On peut associer à chaque IP ce qu'on appelle un nom d'hôte (`hostname`). C'est un nom en toutes lettres plus facile à mémoriser et qui revient exactement au même que d'écrire l'adresse IP, comme le suggère la figure suivante.









Chaque serveur peut ainsi avoir un nom d'hôte plus facile à retenir. Je retiens mieux le nom de notre serveur (`lisa.simple-it.fr`) que son équivalent en adresse IP. :-)

Convertir une IP en nom d'hôte et inversement

Il existe une commande qui est capable d'effectuer la conversion dans les deux sens :

- à partir d'une IP on peut avoir le nom d'hôte correspondant ;
- à partir d'un nom d'hôte, on peut avoir l'IP correspondante.

Cette commande, c'est `host`. Donnez-lui en paramètre une IP ou un nom d'hôte.

Par exemple :

Code : Console

```
$ host siteduzero.com
siteduzero.com has address 92.243.25.239
siteduzero.com mail is handled by 0 mail.siteduzero.com
```

La commande nous répond que l'IP de `siteduzero.com` est `92.243.25.239`. Elle nous indique par ailleurs le nom du serveur qui gère les e-mails.

Maintenant, essayons à l'envers avec l'IP :

Code : Console

```
$ host 92.243.25.239
123.219.248.80.in-addr.arpa domain name pointer lisa.simple-it.fr.
```

On nous répond que le nom d'hôte de `92.243.25.239` est `lisa.simple-it.fr`.


Mais, je croyais que c'était `siteduzero.com` cette IP ?

Oui, en fait il s'agit d'un synonyme dans le cas présent :\\`siteduzero.com = lisa.simple-it.fr`.

Vous pouvez essayer la même manipulation avec d'autres IP et noms d'hôte : prenez des sites que vous connaissez comme par exemple `mozilla.org`, `google.fr`, etc.

Gérer les noms d'hôte personnalisés

Les associations entre les IP et les noms d'hôte sont faites sur ce que l'on appelle des *serveurs DNS*. Nous n'allons pas entrer dans le détail, mais sachez en gros que chaque fournisseur d'accès met en place des serveurs DNS qui fournissent la liste des correspondances IP ↔ noms d'hôte.
Si vous voulez en découvrir plus sur le fonctionnement des DNS, je vous invite à lire mon tutoriel sur les DNS.

Ainsi, lorsque vous tapez `siteduzero.com` dans votre navigateur, vous pouvez obtenir l'adresse IP correspondante et naviguer sur le Site du Zéro.
C'est quand même plus pratique que d'avoir à retenir l'IP !

Vous ne pouvez pas modifier la liste des correspondances IP ↔ noms d'hôte sur le serveur DNS (puisque ce serveur est utilisé par de nombreuses personnes), mais en revanche vous pouvez établir une liste de correspondances personnalisée sur votre ordinateur.

Ouvrez pour cela en root le fichier `/etc/hosts` :

Code : Console

```
$ sudo nano /etc/hosts
```

Dedans, vous devriez avoir des lignes ressemblant à ceci :

Code : Console

```
127.0.0.1       localhost
127.0.1.1       mateo21-laptop
```

À gauche l'IP, à droite le nom d'hôte correspondant. Écrire `localhost` est donc équivalent à écrire `127.0.0.1`.

Vous pouvez ajouter des lignes sur le même modèle pour faire correspondre une IP à un nom d'hôte.

Quel intérêt ? Cela dépend. Parfois, les DNS ne fonctionnent pas bien pendant de courtes périodes (c'est très rare, mais ça peut arriver). Dans ce cas, il est plus simple de modifier votre fichier `hosts` pour pouvoir continuer à consulter votre site préféré en « forçant » l'association du nom d'hôte et de l'IP.

Vous pourriez donc ajouter :

Code : Console

```
92.243.25.239      siteduzero.com
```

Enregistrez, ouvrez un navigateur, puis tapez `siteduzero.com` pour voir si ça fonctionne.


Attention : cette technique a l'avantage de forcer l'association, mais si notre serveur change un jour d'IP, votre ordinateur ne sera pas au courant ! En règle générale, il est préférable d'utiliser les serveurs DNS qui se mettent

 régulièrement à jour (une fois par jour, en moyenne) afin d'avoir toujours une liste actualisée.

Sur un réseau local, il peut être pratique d'associer un nom d'hôte à chaque PC pour pouvoir vous y connecter sans avoir à retenir l'IP :

Code : Console

```
192.168.0.5      pc-papa
```

Ainsi, écrire `pc-papa` vous permet d'accéder à cet ordinateur sans avoir à retenir l'adresse IP correspondante.

`whois` : tout savoir sur un nom de domaine

Chaque nom de domaine doit obligatoirement indiquer qui se trouve derrière : nom, prénom, adresse et moyens de contact. C'est une règle.

L'outil `whois` vous permet d'obtenir facilement ces informations pour n'importe quel nom de domaine :

Code : Console

```
$ whois siteduzero.com

[...]

domain: siteduzero.com
reg_created: 2002-06-09 21:53:29
expires: 2011-06-09 21:53:29
created: 2007-02-27 06:56:43
changed: 2010-04-13 15:35:32
transfer-prohibited: yes
ns0: a.dns.gandi.net
ns1: b.dns.gandi.net
ns2: c.dns.gandi.net
owner-c:
  nic-hdl: PD2500-GANDI
  owner-name: Simple IT SARL
  organisation: Simple IT SARL
  person: Pierre DUBUC
  address: 23 Rue Le Peletier
  zipcode: 75009
  city: Paris
  country: France
lastupdated: 2010-05-17 10:27:41

[...]
```

Utilisez ces informations avec parcimonie. En général, on y a recours lorsque l'on a besoin de contacter le propriétaire d'un nom de domaine ou d'une adresse IP, pour régler un litige mettant en jeu le nom de domaine ou l'IP en question par exemple.

ifconfig & netstat : gérer et analyser le trafic réseau

Nous allons découvrir ici deux commandes : `ifconfig` et `netstat`. La première permet de gérer les connexions réseau de votre machine (pour les activer / désactiver, par exemple) tandis que la seconde vous permet d'analyser ces connexions, de connaître des statistiques, etc.

`ifconfig` : liste des interfaces réseau

Votre ordinateur possède en général plusieurs **interfaces réseau**, c'est-à-dire plusieurs moyens de se connecter au réseau.

Tapez `ifconfig` dans la console pour voir ce que ça donne :

Code : Console

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:90:f5:56:44:5a
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          Packets reçus:0 erreurs:0 :0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          Octets reçus:0 (0.0 B) Octets transmis:0 (0.0 B)
          Interruption:220 Adresse de base:0xe000

lo        Link encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          adr inet6: ::1/128 Scope:Hôte
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          Packets reçus:10 erreurs:0 :0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          Octets reçus:500 (500.0 B) Octets transmis:500 (500.0 B)

wlan0     Link encap:Ethernet  HWaddr 00:19:d2:61:90:0a
          inet adr:192.168.1.2  Bcast:192.168.1.255  Masque:255.255.255.0
          adr inet6: fe80::219:d2ff:fe61:900a/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Packets reçus:5238 erreurs:0 :0 overruns:0 frame:0
          TX packets:4899 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          Octets reçus:5069449 (5.0 MB) Octets transmis:1202459 (1.2 MB)
```

On distingue ici trois interfaces réseau. Vous en avez peut-être plus, peut-être moins ; tout dépend de votre ordinateur.

Les interfaces que j'ai sont assez courantes, détaillons-les :

- `eth0` : cela correspond à la connexion par câble réseau (ce qu'on appelle en général le *câble RJ45* – figure suivante). Si votre PC est relié au réseau via un câble, c'est sûrement ce moyen de communication que vous utilisez actuellement. Notez que certains ordinateurs (et notamment les serveurs) ont plusieurs sorties réseau filaires. Dans ce cas, vous devriez voir aussi des interfaces `eth1`, `eth2`, etc.
- `lo` : c'est la boucle locale. Tout le monde devrait avoir cette interface. Elle correspond à une connexion à… vous-mêmes. C'est pour cela qu'on l'appelle la boucle locale : tout ce qui est envoyé par là vous revient automatiquement. Cela peut paraître inutile, mais on a parfois besoin de se connecter à soi-même pour des raisons pratiques.
- `wlan0` : il s'agit d'une connexion sans-fil type Wi-Fi. Là encore, bien que ce soit plus rare, si vous avez plusieurs cartes réseau sans fil, vous aurez un `wlan1`, `wlan2`, etc.



Observez les résultats de ma commande et essayez de deviner par quelle interface réseau je me connecte à l'internet.

…

Vous avez trouvé ? Il ne fallait pas avoir peur de lire le détail des messages.
En effet, bien que je possède une sortie réseau filaire (RJ45), j'utilise ici le Wi-Fi, comme en témoigne la ligne `Packets reçus:5238` pour le Wi-Fi `wlan0` (alors qu'il y en a 0 pour `eth0`). C'est donc l'interface active que j'utilise le plus.

La commande `ifconfig` permet aussi de faire des réglages réseau. Toutefois, cela sortirait un peu du cadre de ce cours et il vous faudrait des connaissances en réseau pour bien l'utiliser.
Voici cependant un réglage très simple que vous pouvez faire et qui vous sera probablement utile : l'activation / désactivation d'interface.

Il suffit d'écrire une commande sous cette forme :

Code : Console

```
ifconfig interface etat
```

Remplacez :

- `interface` par le nom de l'interface que vous voulez modifier (`eth0`, `wlan0`...) ;
- `etat` par `up` ou `down` selon si vous voulez activer ou désactiver l'interface.

Exemple :

Code : Console

```
$ ifconfig eth0 down
```

... désactive l'interface `eth0` (filaire). Plus aucun trafic ne pourra alors circuler par l'interface `eth0`.

Code : Console

```
$ ifconfig eth0 up
```

... la réactive de nouveau.

Vous aurez peut-être besoin de connaître ces commandes un jour ou l'autre si vous devez désactiver puis réactiver une interface pour prendre en compte des changements dans la configuration de votre réseau.

`netstat` : statistiques sur le réseau

La commande `netstat` risque de vous paraître un peu complexe si vous avez peu de connaissances concernant les réseaux, mais elle est incontournable quand on veut savoir ce que notre machine est en train de faire sur le réseau.

`netstat` peut afficher beaucoup d'informations. Pour sélectionner celles qui nous intéressent, on a recours à de nombreux paramètres.
Plutôt que de les expliquer un par un, je vais vous montrer quelques combinaisons de paramètres qui donnent des résultats intéressants.

`netstat -i` : *statistiques des interfaces réseau*

Pour commencer, essayez l'option `-i` :

Code : Console

```
$ netstat -i
Table d'interfaces noyau
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500 0         0      0      0 0            0      0      0      0 BMU
lo        16436 0        10      0      0 0           10      0      0      0 LRU
wlan0      1500 0      5161      0      0 0         4810      0      0      0 BMR
```

Vous n'aurez pas nécessairement les mêmes lignes que moi ; tout dépend de votre ordinateur.
Il s'agit là d'un tableau présentant, pour chaque interface réseau que vous avez, une série de statistiques d'utilisation. On retrouve ici nos interfaces `eth0`, `lo` et `wlan0`.

Comme vous le voyez sur la colonne `RX-ERR`, c'est `wlan0` qui est l'interface la plus active. Et vous noterez que `lo` est un petit peu utilisée elle aussi ; comme quoi se connecter à soi-même peut s'avérer utile.

Je ne rentrerai pas dans le détail de ces colonnes car c'est assez technique, mais vous savez au moins détecter l'activité de vos interfaces grâce à cette commande.

`netstat -uta` : lister toutes les connexions ouvertes

Code : Console

```
$ netstat -uta
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 mateo21-laptop.lo:60997 debian-mirror.mirro:ftp ESTABLISHE
tcp        1      0 mateo21-laptop.lo:33721 lisa.simple-it.fr:www CLOSE_WAIT
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
udp        0      0 *:bootpc                *:*
udp        0      0 *:mdns                  *:*
udp        0      0 *:45176                 *:*
```

Les options signifient :

- `-u` : afficher les connexions UDP ;
- `-t` : afficher les connexions TCP ;
- `-a` : afficher toutes les connexions quel que soit leur état.

TCP et UDP sont deux protocoles différents pour envoyer des données sur le réseau.
UDP est plutôt utilisé dans les jeux en réseau et pour les communications vocales (avec Skype, par exemple). Sinon, de manière générale, TCP est le protocole le plus utilisé. Je n'irai pas plus loin dans les explications mais vous pouvez vous renseigner si le sujet vous intéresse.

Pour filtrer un peu, on va enlever les connexions UDP qui, la plupart du temps, sont moins importantes :

Code : Console

```
$ netstat -ta
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 mateo21-laptop.lo:60997 debian-mirror.mirro:ftp ESTABLISHE
tcp        0   4107 mateo21-laptop.lo:33721 lisa.simple-it.fr:www ESTABLISHED
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
```

Ce tableau vous indique qui, depuis l'adresse locale, est connecté à qui (à une adresse distante).

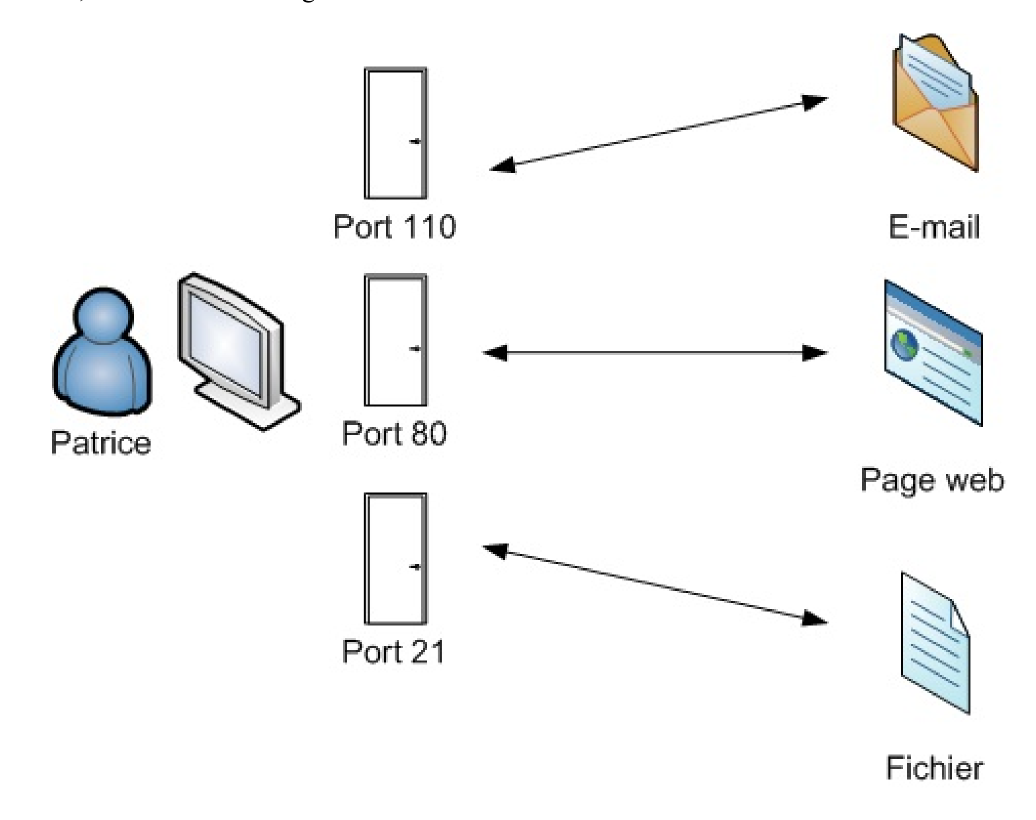
Chaque connexion a un état. Ici, on repère les états `LISTEN` et `ESTABLISHED`.
De nombreux états sont possibles ; en voici quelques-uns à connaître :

- `ESTABLISHED` : la connexion a été établie avec l'ordinateur distant ;
- `TIME_WAIT` : la connexion attend le traitement de tous les paquets encore sur le réseau avant de commencer la fermeture ;

- `CLOSE_WAIT` : le serveur distant a arrêté la connexion de lui-même (peut-être parce que vous êtes restés inactifs trop longtemps ?) ;
- `CLOSED` : la connexion n'est pas utilisée ;
- `CLOSING` : la fermeture de la connexion est entamée mais toutes les données n'ont pas encore été envoyées ;
- `LISTEN` : à l'écoute des connexions entrantes.

Il y en a d'autres que vous pouvez lire dans la documentation. Globalement, ce qu'il faut retenir, c'est que les connexions à l'état `LISTEN` ne sont pas utilisées actuellement mais qu'elles « écoutent » le réseau au cas où quelqu'un veuille se connecter à votre ordinateur.

Regardez en particulier le **port** sur lequel ces connexions écoutent (après le symbole « : ») car c'est probablement l'information la plus intéressante. En effet, on peut se connecter à chaque ordinateur via différentes « portes » appelées *ports*. Chaque service utilise un port différent, comme l'illustre la figure suivante.



À la première ligne, vous avez `*:ssh`, ce qui signifie que SSH est en train d'écouter sur le port de SSH au cas où quelqu'un veuille se connecter à votre machine. C'est logique puisque j'ai activé le serveur SSH pour pouvoir m'y connecter à distance au besoin.

D'autres connexions, elles, sont déjà établies et donc en cours d'utilisation. Par exemple, au niveau de l'adresse distante, je suis connecté par FTP à `debian-mirror.mirro:ftp` et je suis connecté à un serveur web `lisa.simple-it.fr:www`. En clair, je suis en train de charger une page sur le Site du Zéro. ;)

Vous pouvez ajouter `-n` si vous désirez avoir les numéros des ports plutôt qu'une description en toutes lettres :

Code : Console

```
$ netstat -tan
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp       15      0 192.168.1.2:60997       128.101.240.212:21      CLOSE_WAIT
tcp        0      0 192.168.1.2:54001       80.248.219.123:80       ESTABLISHE
tcp6       0      0 :::22                   :::*                    LISTEN
```

Cela correspond aux ports que l'on connaît : 22 pour SSH, 21 pour FTP, 80 pour le web, etc.

`netstat -lt` : liste des connexions en état d'écoute

Très utile, l'option `-l` vous permet de filtrer les connexions à l'état `LISTEN` et donc de savoir quels ports de serveur sont susceptibles d'être utilisés en ce moment sur votre machine.

Code : Console

```
$ netstat -lt
Connexions Internet actives (seulement serveurs)
Proto Recv-Q Send-Q Adresse locale           Adresse distante        Etat
tcp        0      0 *:ssh                    *:*                     LISTEN
tcp        0      0 localhost:ipp            *:*                     LISTEN
tcp6       0      0 [::]:ssh                 [::]:*                  LISTEN
```

`netstat -s` : statistiques résumées

Enfin, si vous êtes très friands de statistiques réseau, `-s` est fait pour vous :

Code : Console

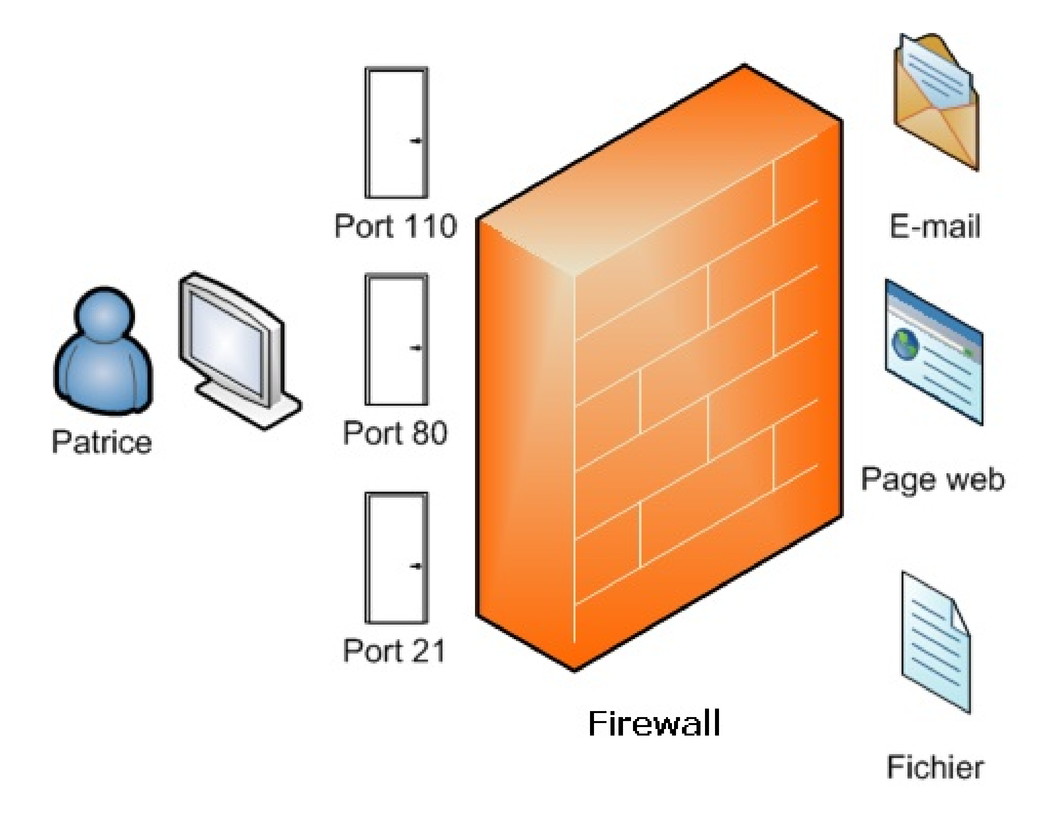
```
$ netstat -s
Ip:
    7443 paquets reçus au total
    1 avec des en-têtes invalides
    8 avec des adresses invalides
    0 réacheminés
    0 paquets arrivant rejetés
    7354 paquets entrants délivrés
    7226 requêtes envoyées
Icmp:
    0 Messages ICMP reçus
    0 messages ICMP entrant échoués

[...]
```

iptables : le pare-feu de référence

Maintenant que nous savons analyser le trafic réseau et ainsi voir un peu ce qui se passe, nous allons nous atteler au filtrage du trafic à l'aide d'un pare-feu.

Le plus célèbre pare-feu utilisé sous Linux est `iptables`. Il permet d'établir un certain nombre de **règles** pour dire par quels ports *on peut* se connecter à votre ordinateur, mais aussi à quels ports vous *avez le droit* de vous connecter (figure suivante). On peut également filtrer par IP, mais nous ne détaillerons pas cela ici.



Par exemple, si je veux empêcher toute connexion FTP (parce que je considère que le FTP n'est pas sûr), je peux souhaiter bloquer le port 21 (utilisé par FTP).
En général la technique ne consiste pas à bloquer certains ports mais plutôt à bloquer par défaut **tous** les ports et à en autoriser seulement quelques-uns.



Attends… c'est quoi le but, exactement ? Bloquer tout le trafic réseau ? Pour quoi faire ?

C'est avant tout une question de sécurité. Le but d'un pare-feu est d'empêcher que des programmes puissent communiquer sur le réseau sans votre accord. Aujourd'hui, même sous Windows (depuis Windows XP SP2), un pare-feu est intégré par défaut, tant le problème est important.

Avoir un pare-feu ne vous prémunit pas contre les virus (bien que sous Linux, ils restent rares). En revanche, cela rend la tâche **particulièrement difficile** aux pirates qui voudraient accéder à votre machine.

Vous vous souvenez de ce que je vous ai expliqué un peu plus tôt ? Chaque ordinateur possède plusieurs portes d'entrée possibles.
Notre objectif est de bloquer par défaut toutes ces portes et d'autoriser seulement celles dont vous avez besoin, que vous considérez comme « sûres » et que vous utilisez. Par exemple, le port 80 utilisé pour le web est un port sûr que vous pouvez activer.

Notez, et c'est important, qu'il y a des portes d'entrée et des portes de sortie sur votre ordinateur (ce ne sont pas nécessairement les mêmes).



`iptables` est un programme extrêmement puissant, mais tout aussi complexe. Nous ne verrons que des fonctionnalités basiques (et ce sera déjà pas mal ;)). Sachez qu'il peut faire bien plus que ce que l'on va voir : pour en savoir plus, comme d'habitude, lisez le manuel.

`iptables` s'utilise en « root »

Pour manipuler `iptables`, vous devez impérativement être en « root ». Pour la suite des opérations, je vous recommande donc de passer en superutilisateur dès à présent :

Code : Console

```
$ sudo su
```

iptables -L : afficher les règles

Avec `iptables -L` (attention, un « L » majuscule), vous pouvez afficher les règles qui régissent actuellement le pare-feu :

Code : Console

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

On repère trois sections :

- `Chain INPUT` : correspond aux règles manipulant le trafic entrant ;
- `Chain FORWARD` : correspond aux règles manipulant la redirection du trafic ;
- `Chain OUTPUT` : correspond aux règles manipulant le trafic sortant.

Nous ne verrons pas ici la section `FORWARD`. `iptables` permet de rediriger le trafic, mais c'est assez compliqué et ne nous intéresse pas ici. Nous aurons déjà suffisamment de quoi faire avec `INPUT` et `OUTPUT`.

Actuellement, chez moi, les règles sont vides. Il y a trois tableaux mais qui ne contiennent aucune ligne. Par ailleurs, vous noterez à chaque fois les mots `(policy ACCEPT)` qui signifient que, par défaut, tout le trafic est accepté. Donc chez moi, pour le moment, le pare-feu est tout simplement inactif car il ne bloque rien ; mon ordinateur est une vraie passoire. :-D

Si vous avez déjà des règles inscrites dans votre pare-feu (ce qui ne devrait pas être votre cas, mais on ne sait jamais), sachez que vous pouvez les réinitialiser.
Ne le faites que **si vous êtes certains de vouloir le faire**. En effet, sur un ordinateur partagé, peut-être quelqu'un a-t-il déjà configuré le pare-feu et il serait dommage de saboter tout son travail.

Code : Console

```
# iptables -F   <-- Attention ! Réinitialise toutes les règles iptables !
```

Le principe des règles

Voici ce que cela pourrait donner lorsqu'on aura établi des règles, par exemple ici pour la section `INPUT` :

Code : Console

```
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
```

```
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:www
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --
  anywhere             anywhere            tcp dpt:imap2
```

Première chose à savoir : **l'ordre des règles est important**. En effet, `iptables` les lit de haut en bas et la position de ces règles influe sur le résultat final. Sachez donc que les règles sont numérotées.

Pour avoir les numéros, ajoutez `--line-numbers` :

Code : Console

```
# iptables -L --line-numbers
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     tcp  --
  anywhere             anywhere            tcp dpt:www
2    ACCEPT     tcp  --
  anywhere             anywhere            tcp dpt:ssh
3    ACCEPT     tcp  --
  anywhere             anywhere            tcp dpt:imap2
```

Ainsi, la règle filtrant SSH est la règle n° 2.

Chaque ligne correspond à une règle différente qui permet de filtrer ou non une IP ou un port. Parmi les colonnes intéressantes, on note :

- `target` : ce que fait la règle. Ici c'est `ACCEPT`, c'est-à-dire que cette ligne autorise un port et / ou une IP ;
- `prot` : le protocole utilisé (`tcp`, `udp`, `icmp`). Je rappelle que TCP est celui auquel on a le plus recourt. ICMP permet à votre ordinateur de répondre aux requêtes de type « ping » ;
- `source` : l'IP de source. Pour `INPUT`, la source est l'ordinateur distant qui se connecte à vous ;
- `destination` : l'IP de destination. Pour `OUTPUT`, c'est l'ordinateur auquel on se connecte ;
- *la dernière colonne* : elle indique le port après les deux points « : ». Ce port est affiché en toutes lettres, mais avec `-n` vous pouvez obtenir le numéro correspondant.

Sur mon exemple, seuls les ports web, `ssh` et `imap2` (e-mail) sont autorisés en entrée. Personne ne peut se connecter à la machine par un autre biais.
En effet, si vous regardez bien, par défaut j'ai configuré le pare-feu pour qu'il ignore tous les autres paquets : `(policy DROP)`.

Nous allons maintenant apprendre à faire tout cela.

Ajouter et supprimer des règles

Voici les principales commandes à connaître.

- `-A chain` : ajoute une règle en fin de liste pour la `chain` indiquée (`INPUT` ou `OUTPUT`, par exemple).
- `-D chain rulenum` : supprime la règle n° `rulenum` pour la `chain` indiquée.
- `-I chain rulenum` : insère une règle au milieu de la liste à la position indiquée par `rulenum`. Si vous n'indiquez pas de position `rulenum`, la règle sera insérée en premier, tout en haut dans la liste.
- `-R chain rulenum` : remplace la règle n° `rulenum` dans la `chain` indiquée.
- `-L` : liste les règles (nous l'avons déjà vu).
- `-F chain` : vide toutes les règles de la `chain` indiquée. Cela revient à supprimer toutes les règles une par une pour cette `chain`.
- `-P chain regle` : modifie la règle par défaut pour la `chain`. Cela permet de dire, par exemple, que par défaut tous les ports sont fermés, sauf ceux que l'on a indiqués dans les règles.

De manière générale, l'ajout d'une règle se passe suivant ce schéma :

Code : Console

```
iptables -A (chain) -p (protocole) --dport (port) -j (décision)
```

Remplacez `chain` par la section qui vous intéresse (`INPUT` ou `OUTPUT`), `protocole` par le nom du protocole à filtrer (TCP, UDP, ICMP…) et enfin `décision` par la décision à prendre : `ACCEPT` pour accepter le paquet, `REJECT` pour le rejeter ou bien `DROP` pour l'ignorer complètement.

Le mieux est de découvrir comment on ajoute une règle par une série d'exemples. ;)

Code : Console

```
# iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

Cela ajoute à la section `INPUT` (donc, pour le trafic entrant) une règle sur les données reçues via le protocole TCP sur le port de `ssh` (vous pouvez remplacer `ssh` par le numéro du port, soit 22). Lorsque votre ordinateur recevra des données en TCP sur le port de SSH, celles-ci seront acceptées ; cela vous permettra donc de vous connecter à distance à votre PC via SSH.

Vous pouvez faire de même avec d'autres ports :

Code : Console

```
# iptables -A INPUT -p tcp --dport www -j ACCEPT
```

… pour le web (80).

Code : Console

```
# iptables -A INPUT -p tcp --dport imap2 -j ACCEPT
```

… pour les mails, etc.



Si vous ne précisez pas de port (en omettant la section `dport`), tous les ports seront acceptés !

Autoriser les pings

En plus d'autoriser le trafic sur ces ports, je peux vous conseiller d'autoriser le protocole ICMP (pour pouvoir faire un ping) sur tous ces derniers :

Code : Console

```
# iptables -A INPUT -p icmp -j ACCEPT
```

Comme je n'ai pas indiqué de section `--dport`, cette règle s'applique à tous les ports, **mais pour les pings (icmp) uniquement** !

Votre ordinateur répondra alors aux « pings » pour indiquer qu'il est bien en vie.

Vos règles `iptables` pour `INPUT` devraient maintenant ressembler à ceci :

Code : Console

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:www
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --
  anywhere             anywhere            tcp dpt:imap2
ACCEPT     icmp --  anywhere             anywhere
```

Autoriser les connexions locales et déjà ouvertes

Pour l'instant, nos règles sont encore un peu trop restrictives et pas vraiment utilisables (vous risquez de ne plus pouvoir faire grand-chose).
Je vous propose d'ajouter deux règles pour « assouplir » un peu votre pare-feu et le rendre enfin utilisable.

Code : Console

```
# iptables -A INPUT -i lo -j ACCEPT
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Ces deux règles utilisent des options un peu différentes de celles que nous avons vues jusqu'ici. Voici quelques explications.

1. La première règle autorise tout le trafic sur l'interface de loopback locale grâce à `-i lo`. Il n'y a pas de risque à autoriser votre ordinateur à communiquer avec lui-même, d'autant plus qu'il en a parfois besoin !
2. La seconde règle autorise toutes les connexions qui sont déjà à l'état `ESTABLISHED` ou `RELATED`. En clair, elle autorise toutes les connexions qui ont été demandées par votre PC. Là encore, cela permet d'assouplir le pare-feu et de le rendre fonctionnel pour une utilisation quotidienne.

Refuser toutes les autres connexions par défaut

Il reste un point essentiel à traiter car, **pour l'instant, ce filtrage ne sert à rien**. En effet, nous avons indiqué quelles données nous autorisions, mais **nous n'avons pas dit que toutes les autres devaient être refusées** !

Changez donc la règle par défaut pour `DROP` par exemple :

Code : Console

```
# iptables -P INPUT DROP
```

`iptables` devrait maintenant indiquer que par défaut tout est refusé, sauf ce qui est indiqué par les lignes dans le tableau :

Code : Console

```
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:www
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --
  anywhere             anywhere            tcp dpt:imap2
ACCEPT     icmp --  anywhere             anywhere
```

Le filtrage est radical. Nous n'avons pas autorisé beaucoup de ports et il se pourrait que vous vous rendiez compte que certaines applications n'arrivent plus à accéder à l'internet (normal, leur port doit être filtré).

À vous de savoir quels ports ces applications utilisent pour modifier les règles en conséquence.
Au besoin, pensez à faire de même pour les règles de sortie (`OUTPUT`).

Appliquer les règles au démarrage

Si vous redémarrez votre ordinateur, les règles `iptables` auront disparu !
Le seul moyen pour qu'elles soient chargées au démarrage consiste à créer un script qui sera exécuté au démarrage.

Justement, ça tombe bien, nous allons étudier la programmation de scripts shell sous Linux dans la prochaine partie. :-)
En attendant, si vous voulez lire un mode d'emploi rapide pour mettre les règles au démarrage, je vous invite à lire la documentation `ubuntu-fr`.



Comme vous avez pu le constater, `iptables` est donc un pare-feu assez compliqué. Sachez que des développeurs ont travaillé sur un programme qui simplifie l'utilisation d'`iptables` : **ufw** (*Uncomplicated Firewall*). Contrairement à `iptables`, ce programme n'est pas disponible partout, mais on le trouve dans les versions récentes d'Ubuntu.

En résumé

- Sur l'internet, chaque ordinateur est identifié par une adresse IP. Par exemple : `86.172.120.28`.
- On peut associer à chaque adresse IP un nom d'hôte, plus facile à retenir, comme `lisa.simple-it.fr`. Écrire le nom d'hôte est équivalent à écrire l'adresse IP.
- La commande `host` permet de traduire une IP en nom d'hôte et inversement.
- `ifconfig` liste les interfaces réseau (cartes réseau) de votre machine et permet de les configurer ainsi que de les activer.
- `netstat` affiche la liste des connexions ouvertes sur votre machine. Elle indique notamment quel port est utilisé à chaque fois, le port représentant en quelque sorte la porte d'entrée à votre machine.
- Il est possible de bloquer l'accès à certains ports avec le programme `iptables`, un pare-feu (*firewall*) très puissant. Celui-ci est cependant assez complexe à configurer.

 Compiler un programme depuis les sources

Nous avons découvert dans un chapitre précédent combien il était facile d'installer de nouveaux programmes sous Ubuntu à l'aide de la commande `apt-get`. Cette technique permet de télécharger et installer la grande majorité des programmes.

Cependant, il arrive parfois qu'il soit nécessaire d'installer un programme manuellement car il n'apparaît pas dans `apt-get`. Dans ce cas, il faut récupérer les sources du programme et les compiler soi-même pour créer un exécutable !

Cette opération peut se révéler assez complexe dans certains cas. Il nous faudra télécharger les sources, les extraire d'une archive gzippée et les compiler manuellement. Je vous propose dans ce chapitre de mettre en pratique vos connaissances en découvrant pas à pas la compilation d'un programme.

Essayez d'abord de trouver un paquet .deb

La plupart des programmes dont vous aurez besoin sous Ubuntu sont référencés dans des dépôts et accessibles via une simple commande : `apt-get`. Toutefois, certains programmes récents ou encore en développement ne sont pas disponibles via `apt-get` (c'est le cas des programmes en cours de développement ou de ceux qui ne sont pas encore assez connus pour être intégrés aux dépôts officiels d'Ubuntu.

Dans un tel cas, les choses se corsent nettement. Sous Windows, nous avions l'habitude de nous rendre sur le site web du logiciel et de télécharger le `.exe` d'installation. Or, les développeurs qui écrivent des programmes pour Linux ne créent presque jamais de gestionnaires d'installation comme on en voit sous Windows.



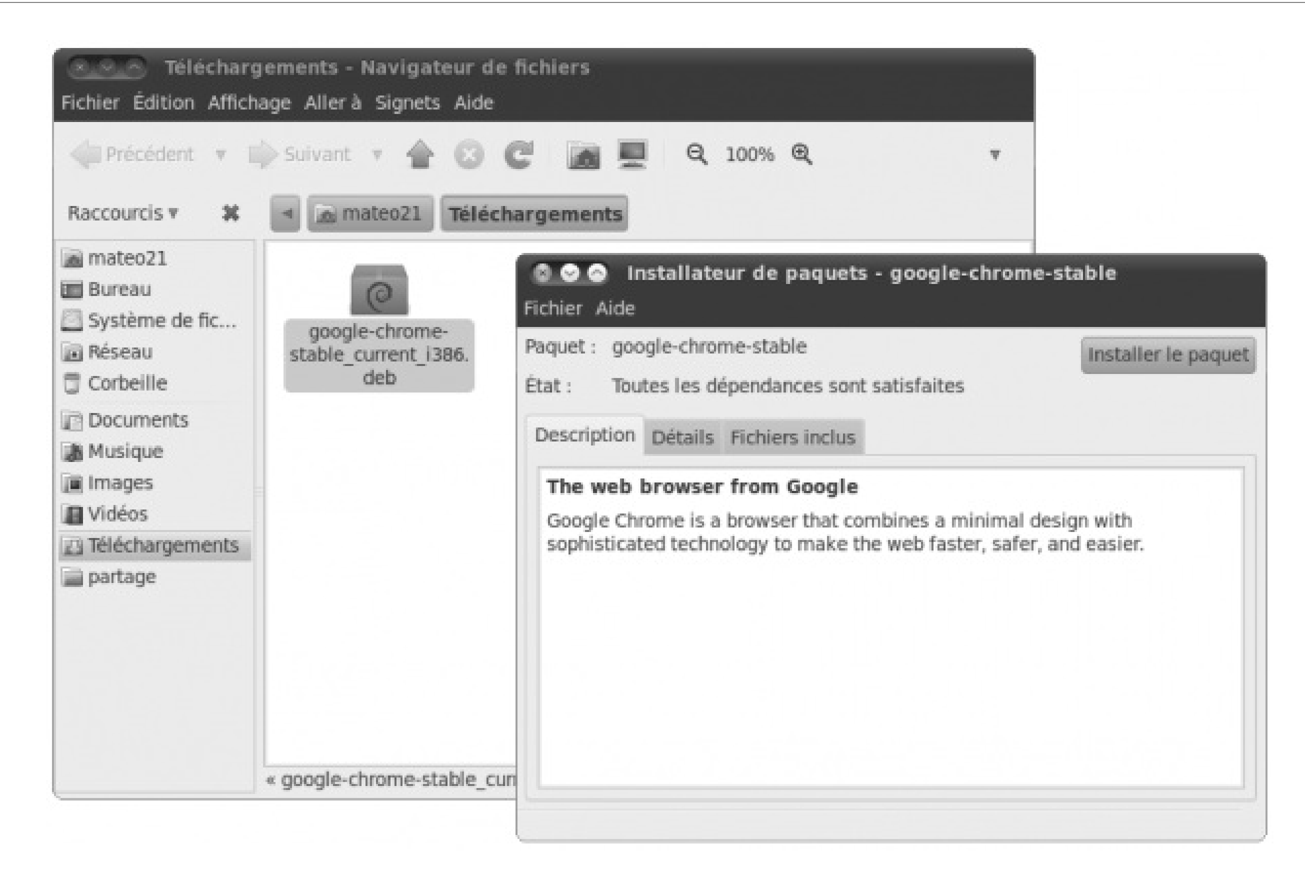
Pourquoi ? Le concept de programme d'installation n'existe-t-il pas sous Linux ?

En fait, ce concept existe mais on l'évite. Pourquoi ? La raison est qu'il existe une grande diversité de distributions Linux et d'architectures d'ordinateurs (selon le type de processeur par exemple). Cette diversité est une des grandes forces de Linux, mais il est du coup presque impossible de proposer un programme d'installation qui convienne à tout le monde et qui pourra s'installer sur toutes les machines. Il faudrait créer autant de programmes d'installation qu'il existe de types de machines différents !

Quand `apt-get` ne propose pas le programme que l'on recherche, il est parfois possible de trouver sur le site web du logiciel un paquetage `.deb`. C'est en quelque sorte l'équivalent du programme d'installation, mais celui-ci est spécifique à Debian et à ses distributions dérivées (dont fait partie Ubuntu). Les `.deb` ne fonctionnent pas sur les distributions utilisant d'autres outils ; Red Hat utilise des `.rpm` par exemple.
Notez que le programme `alien` est capable de convertir un `.rpm` en `.deb` au besoin.

Si, par bonheur, vous trouvez le `.deb` du programme que vous souhaitez installer, téléchargez-le et double-cliquez dessus. Essayons par exemple de récupérer de cette façon Google Chrome sur le site web de Google.

Une fois le `.deb` téléchargé, double-cliquez dessus. Une fenêtre apparaît pour vous proposer d'installer le logiciel (figure suivante).



Si aucune erreur n'apparaît, vous avez de la chance, vous pouvez procéder à l'installation. Sinon, cela signifie :

- soit que vous avez téléchargé un `.deb` ne correspondant pas à votre machine. Vérifiez que vous n'avez pas pris une version 32 bits au lieu de 64 bits (ou inversement) ;
- soit qu'il vous manque des dépendances pour pouvoir installer convenablement le programme. Et là, cela peut vite devenir un casse-tête ! Il faut d'abord installer le programme manquant avant d'aller plus loin.



Souvenez-vous : chaque programme dépend de l'installation d'autres programmes, comme nous l'avions découvert précédemment. Un outil comme `apt-get` permet de télécharger automatiquement les dépendances du programme, ce qui simplifie grandement les choses.

Si même le paquetage `.deb` n'est pas disponible, il ne reste alors qu'une solution : récupérer le *code source* du programme et le compiler soi-même. On peut ainsi créer un exécutable spécialement optimisé pour sa machine.
L'exécutable est l'équivalent du `.exe` de Windows, même s'il n'a en général pas d'extension sous Linux.

Quand il n'y a pas d'autre solution : la compilation

Si le programme que vous recherchez n'est pas dans les dépôts (`apt-get`) et que vous ne parvenez pas non plus à trouver de `.deb` prêt à l'emploi sur le web, vous allez devoir le compiler depuis ses sources.

Qu'est-ce que la compilation ?

La **compilation** est un procédé qui permet de transformer le code source d'un programme en un exécutable que l'on peut utiliser. Le code source correspond en quelque sorte aux ingrédients d'une recette (les œufs, la farine…) et l'exécutable au gâteau final. Dans cette métaphore, la compilation correspondrait à la cuisson du gâteau. :-)

Étant donné que la plupart des programmes sous Linux sont libres, nous avons la chance de pouvoir récupérer leurs sources et donc de pouvoir en compiler une version propre à notre machine.

Les étapes de la compilation peuvent varier d'un programme à un autre. Certains sont assez complexes et nécessitent plusieurs préparatifs. Dans ce cas, il faut suivre les instructions indiquées sur le site web du logiciel pour savoir comment compiler (instructions qui sont, bien souvent, en anglais).

Compilation d'un programme pas à pas

Pour compiler des programmes, vous aurez besoin avant toute chose d'installer les outils de compilation. Pour cela, rien de plus simple, il suffit d'installer le paquet `build-essential` :

Code : Console

```
sudo apt-get install build-essential
```

Ceci étant fait, nous pouvons à présent nous intéresser à la compilation proprement dite.

Ici, je vous propose d'apprendre à compiler un petit programme assez simple : `htop`. Il s'agit d'un outil alternatif à `top`, qui permet de voir la liste des programmes en cours d'exécution. Cela sera l'occasion de découvrir les principales commandes de compilation qui vous serviront pour installer la plupart des logiciels.
Notez qu'on le retrouve dans les dépôts via `apt-get`, mais nous allons tout de même essayer de le compiler manuellement pour nous entraîner.

La première étape consiste à se rendre sur le site web du logiciel htop. Une recherche sur le web devrait vous y amener rapidement.

À partir de là, il est indispensable de savoir lire l'anglais. Recherchez sur le site la section « Downloads », puis, sur la page des téléchargements, recherchez les sources. Vous devriez finalement arriver sur une page qui vous propose de télécharger les dernières sources du programme.

Vous allez télécharger une archive compressée `.tar.gz`. Vous connaissez la commande pour extraire ce type d'archive, alors allez-y !

Code : Console

```
tar zxvf htop-0.8.3.tar.gz
```

On peut maintenant se rendre dans le dossier où les fichiers sources ont été décompressés :

Code : Console

```
cd htop-0.8.3
```

Si vous listez le contenu de ce répertoire, vous allez être surpris : il y a beaucoup de fichiers ! Heureusement, vous n'avez pas à vous en préoccuper.

Pour le moment, un seul programme nous intéresse : `configure`. Exécutez-le comme suit :

Code : Console

```
./configure
```

`configure` est un programme qui analyse votre ordinateur et qui vérifie si tous les outils nécessaires à la compilation du logiciel que vous souhaitez installer sont bien présents. Son exécution peut prendre du temps car il effectue de nombreux tests :

Code : Console

```
$ ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
```

```
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3

[...]
```

Un des premiers éléments qu'il va vérifier est la présence du compilateur (`checking for gcc...`) que vous avez normalement dû installer un peu plus tôt avec le paquet `build-essential`.

Patientez le temps de l'exécution de `configure`. Celui-ci va vous indiquer si tout est prêt pour une compilation ou non.

Malheureusement, il arrivera fréquemment que `configure` affiche une erreur en raison d'un manque de dépendances. Dans notre cas, il devrait afficher une erreur comme celle-ci :

Code : Console

```
checking for sys/time.h... yes
checking for unistd.h... (cached) yes
checking curses.h usability... no
checking curses.h presence... no
checking for curses.h... no
configure: error: missing headers:  curses.h
```

L'erreur (sur la dernière ligne) indique en anglais « `missing headers:  curses.h` ». C'est là que les choses se corsent : il faut installer l'élément manquant, en l'occurrence ces fameux *headers* de `curses.h`. Si vous n'êtes pas programmeurs, vous n'avez probablement aucune idée de ce dont il s'agit.

La technique la plus efficace consiste à effectuer une recherche de la ligne d'erreur sur le web, accompagnée de préférence du mot-clé « ubuntu ». Lancez donc une recherche de « configure: error: missing headers: curses.h ubuntu ».

Une recherche de ces mots clés devrait généralement vous afficher des résultats qui vous dirigeront bien souvent sur des forums anglophones.

Il faut alors faire preuve de patience et ne pas avoir peur de lire un peu d'anglais. ;)

L'information à chercher est le nom du *paquet* manquant que vous devez installer. En lisant les forums, vous devriez finir par trouver le nom du paquet que vous recherchez : `libncurses5-dev`. En l'occurrence, il suffit d'installer ce paquet via `apt-get` pour ne plus avoir l'erreur indiquée dans `configure`.

Code : Console

```
sudo apt-get install libncurses5-dev
```

Une fois le paquet installé, relancez `configure` et croisez les doigts pour que l'erreur disparaisse.



Cela n'a pas été scientifiquement prouvé, mais croiser les doigts peut augmenter vos chances de réussite. La magie vaudou est aussi un bon moyen de se sortir des situations compliquées, mais elle est à réserver aux utilisateurs

expérimentés. ;)

Code : Console

```
./configure
```

Si `configure` n'affiche plus la même erreur, vous avez gagné (pour le moment). Il reste maintenant deux possibilités :

- soit vous avez une nouvelle erreur et vous devrez la résoudre de la même manière : en effectuant une recherche sur l'internet pour comprendre ce qui ne va pas. Le plus souvent, il suffira d'installer le paquet manquant avec `apt-get` ;
- soit vous n'avez pas d'erreur et `configure` parvient jusqu'à son terme. Victoire !

Si tout va bien, `configure` n'affichera pas d'erreur. Vous devriez voir s'afficher des lignes similaires à celles-ci :

Code : Console

```
configure: creating ./config.status
config.status: creating plpa-1.1/Makefile
config.status: creating plpa-1.1/src/Makefile
config.status: creating Makefile
config.status: creating htop.1
config.status: creating config.h
config.status: config.h is unchanged
config.status: creating plpa-1.1/src/plpa_config.h
config.status: creating plpa-1.1/src/plpa.h
config.status: executing depfiles commands
```

Le programme est prêt à être compilé ! Rassurez-vous, le plus dur est derrière vous. :-)

Il suffit maintenant de lancer la compilation à l'aide d'une commande toute simple :

Code : Console

```
make
```

Durant la compilation, des lignes barbares s'afficheront dans votre console. Vous ne devriez pas avoir à vous en préoccuper, tous les problèmes ayant normalement été détectés auparavant par `configure`.



La compilation d'un programme peut prendre du temps ; tout dépend de la taille de celui-ci. Il est ainsi bien plus rapide et plus simple de compiler htop que Firefox par exemple.

Une fois la compilation terminée, l'exécutable devrait avoir été créé. Il ne reste plus qu'à l'installer, c'est-à-dire à le copier dans le bon répertoire. Là encore, vous n'avez pas à vous poser beaucoup de questions. Exécutez la commande suivante :

Code : Console

```
sudo make install
```

Il faut être « root » pour cette opération (d'où le `sudo`) car le programme va être copié dans des répertoires système.

Une fois que cela est fait, le programme est installé ! Nous pouvons à présent exécuter htop en tapant le nom de la commande :

Code : Console

```
htop
```

```
CPU[||                        3.9%]   Tasks: 226 total, 1 running
Mem[|||||||||||||||||||230/1002MB]    Load average: 0.08 0.21 0.18
Swp[                      0/567MB]    Uptime: 08:38:17

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
11818 mateo21    20   0  2576  1280   992 R  1.0  0.1  0:07.87 htop
 1147 root       20   0  3608  1328  1148 S  0.0  0.1  0:08.59 hald-addon-storage
 1013 root       20   0 49880 24488  8988 S  0.0  2.4  0:19.53 /usr/bin/X :0 -nr
 6321 mateo21    20   0  248M 40968 26784 S  0.0  4.0  0:01.79 /opt/google/chrome
 6359 mateo21    20   0  112M 28512 13368 S  0.0  2.8  0:02.05 /opt/google/chrome
 6346 mateo21    20   0  120M 18064 10776 S  0.0  1.8  0:00.42 /opt/google/chrome
 1467 mateo21    20   0 50080 13396 10228 S  0.0  1.3  0:11.67 gnome-terminal
 6355 mateo21    20   0  248M 40968 26784 S  0.0  4.0  0:00.59 /opt/google/chrome
 1290 mateo21    20   0 97472 11808  9636 S  0.0  1.2  0:01.26 metacity --replace
  778 nobody     20   0 54116  1104   808 S  0.0  0.1  0:00.97 /usr/bin/memcached
 1341 root       20   0  5176   872   604 S  0.0  0.1  0:04.21 udisks-daemon: pol
  837 root       20   0  5976   728   520 S  0.0  0.1  0:00.74 /usr/sbin/VBoxServ
  539 messageb   20   0  3224  1552   784 S  0.0  0.2  0:00.48 dbus-daemon --syst
 1359 mateo21    20   0 16948  2348  1960 S  0.0  0.2  0:01.29 /usr/lib/gvfs/gvfs
 1381 mateo21    20   0 32492 13892 11076 S  0.0  1.4  0:00.62 /usr/lib/gnome-pan
 1283 mateo21    20   0 20668  8196  6640 S  0.0  0.8  0:00.44 gnome-power-manage
 1372 mateo21    20   0 44260 13404 10396 S  0.0  1.3  0:01.27 /usr/lib/gnome-pan
F1Help  F2Setup F3SearchF4InvertF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Si vous souhaitez désinstaller le programme, il suffit d'exécuter cette commande depuis le répertoire où vous l'avez compilé :

Code : Console

```
sudo make uninstall
```



Vous pouvez sans problème supprimer le répertoire contenant les fichiers sources (celui depuis lequel vous avez compilé). Toutefois, il ne sera alors plus possible de lancer la commande de désinstallation.

En résumé

- La plupart des programmes peuvent être installés facilement avec la commande `apt-get`.
- Certains programmes ne peuvent pas être installés via `apt-get` car ils ne sont pas référencés dans les dépôts d'Ubuntu. Dans ce cas, on peut rechercher sur le web un paquet `.deb` du programme, sous réserve qu'il existe.
- Si la solution précédente échoue, on n'a pas d'autre choix que de compiler le programme à partir de ses sources. Cela consiste à effectuer les opérations suivantes dans l'ordre :
    1. télécharger les sources du programme sur le web (souvent archivées au format `.tar.gz`) ;
    2. décompresser l'archive (`tar zxvf archive.tar.gz`) ;
    3. exécuter `./configure` et résoudre les problèmes ;
    4. exécuter `make` pour compiler ;
    5. exécuter `sudo make install` pour installer le programme.

Partie 5 : Automatisez vos tâches avec des scripts Bash

 Vim : l'éditeur de texte du programmeur

Dans cette dernière partie, nous allons réunir toutes les connaissances que nous avons acquises concernant les commandes utilisées sous Linux. Nous allons les combiner et créer ce que l'on appelle **des scripts shell**.

Le scripting shell est un minilangage de programmation intégré à tous les systèmes Linux et qui vous permet d'automatiser des tâches répétitives. Il s'agit d'un élément très puissant du système que vous devez absolument connaître.

Toutefois, pour programmer, il va vous falloir utiliser un éditeur de texte digne de ce nom. Certes, vous connaissez déjà Nano mais comme je vous l'ai dit ce dernier est très basique. Nous l'avons utilisé au départ pour simplifier, mais il est temps à présent de passer à quelque chose de plus complet et de plus puissant : Vim (prononcez « *Vi aille ème* »).

Installer Vim

Sous Linux, deux puissants éditeurs de texte en console sont à connaître.

- **Vim** : il s'agit d'une version améliorée de l'un des plus anciens éditeurs en console : « Vi » (prononcez les lettres en anglais « *Vi aille* »).
  Vim (*VI iMproved*, version améliorée de Vi) est largement répandu et généralement disponible par défaut sur la plupart des OS basés sur Unix, comme Linux.
- **Emacs** : développé par Richard Stallman, le fondateur du projet GNU dont je vous ai parlé au début du livre, cet éditeur concurrent a lui aussi bien des atouts. On le retrouve plus spécifiquement sous Linux mais il est rarement installé par défaut (un petit `apt-get` suffit, toutefois). Il peut être complété par toute une série de plugins qui lui permettent de faire navigateur web, lecteur audio… Bref, c'est un peu un outil à tout faire.

Sachez qu'il est courant que les gens adoptent et défendent bec et ongles l'un ou l'autre de ces éditeurs. Choisir un éditeur de texte sous Linux, c'est en fait un peu comme choisir une religion (oui, je sais : ils sont fous, ces Linuxiens !).



Hou ! là, c'est important alors ! Lequel choisir ?

En fait, rien ne vous empêche d'apprendre à utiliser les deux. Toutefois ces logiciels sont tellement complets qu'il vous faudra du temps pour vous habituer à chacun d'eux.
Dans la pratique, on prend l'habitude d'en choisir un et de s'y tenir : il est donc rare de voir quelqu'un naviguer entre les deux.

Vim ou Emacs ? Emacs ou Vim ?

Tout cela ne répond pas à votre question, je sais. Mais ne comptez pas sur moi pour vous dire « Utilisez celui-là, il est mieux » : des milliers de *trolleurs* le font mieux que moi sur tous les forums du monde. Et je pourrais m'attirer les foudres divines des adorateurs de l'un ou l'autre éditeur si je m'y risquais.

D'ailleurs, vous devriez vous mettre en tête dès maintenant qu'il n'y en a pas un qui soit nul et l'autre génial ; ce sont juste deux conceptions un peu différentes de ce que doit être un éditeur de texte.

Le meilleur conseil que je puisse vous donner est le suivant : **choisissez d'utiliser le même éditeur que votre ami pro de Linux ou votre collègue de bureau**. L'idéal est d'avoir quelqu'un à proximité qui peut régulièrement vous conseiller. Croyez-moi, s'il est bien un conseil qui soit important dans ce chapitre, c'est celui-là.



Et toi, ton éditeur, c'est quoi ?

Je craignais cette question mais il fallait bien qu'elle soit posée un jour…
Pour ma part, je n'ai jamais eu l'occasion de prendre le temps d'apprendre à utiliser Emacs. Le professeur qui m'a initié à Linux était un habitué de Vim (mais il n'a jamais dit du mal d'Emacs, je le jure !).

Je suis donc à mon tour un habitué de Vim et c'est lui que je vous présenterai dans ce livre.

Installer et lancer Vim

Sur la plupart des distributions Linux, Vim est en général installé par défaut. J'ai bien dit **en général**. En effet, rien n'assure que Vim soit installé par défaut sur votre distribution ; après tout, c'est elle qui choisit les programmes initialement installés.

Sous Ubuntu, il faut savoir que ce n'est pas Vim qui est installé mais **Vim-tiny**, une version allégée. Personnellement, elle ne me convient pas ; de plus, elle est limitée en possibilités. Je vous invite donc à installer le vrai Vim complet en tapant :

Code : Console

```
sudo apt-get install vim
```

Vous pourrez alors lancer le logiciel en tapant la commande `vim`.
La commande `vi` fonctionne aussi mais il est recommandé de taper plutôt `vim`.

Vimtutor : le programme qui vous apprend à utiliser Vim !

Pour les nouveaux utilisateurs, Vim intègre un véritable petit tutoriel !
Ce programme peut être lancé en tapant :

Code : Console

```
vimtutor
```

Si vous ne l'avez pas, installez le paquet `vim-common`... mais normalement il devrait déjà être présent sur votre distribution.

En fait, Vimtutor lance simplement Vim en ouvrant un fichier d'aide prédéfini. Cette introduction à Vim est d'ailleurs en français et accessible à tout le monde, aussi je vous invite à l'essayer et à la lire en complément de ce qui suit.

Petit aperçu :

Code : Console

```
===============================================================================
= B i e n v e n u e   dans le   T u t o r i e l   de   V I M  -  Version 1.5.fr.2
===============================================================================

     Vim est un éditeur très puissant qui a trop de commandes pour pouvoir
     toutes les expliquer dans un cours comme celui-ci, qui est conçu pour en
     décrire suffisamment afin de vous permettre d'utiliser simplement Vim.

     Le temps requis pour suivre ce cours est d'environ 25 à 30 minutes, selon
     le temps que vous passerez à expérimenter.  Les commandes utilisées dans
     les leçons modifieront le texte. Faites une copie de ce fichier afin de
     vous entraîner dessus (si vous avez lancé "vimtutor" ceci est déjà une
     copie).

     Il est important de garder en tête que ce cours est conçu pour apprendre
     par la pratique.  Cela signifie que vous devez exécuter les commandes
     pour les apprendre correctement.  Si vous vous contentez de lire le
     texte, vous oublierez les commandes !

     Maintenant, vérifiez que votre clavier n'est PAS verouillé en majuscules,
     et appuyez la touche  j  le nombre de fois suffisant pour que la leçon
     1.1 remplisse complètement l'écran.
```


Il faut compter en général une bonne demi-heure pour suivre le Vimtutor. Cela vous fait une bonne petite introduction au logiciel, mais gardez bien entendu à l'esprit que les possibilités sont bien plus grandes et que vous n'aurez pas tout vu à l'issue de sa lecture.

Les modes d'édition de Vim

Commencez par lancer Vim. Comme je vous l'ai dit plus tôt, il suffit pour cela de taper la commande suivante :

Code : Console

```
vim
```

Vim s'ouvre alors (figure suivante).

```
~
~
~
~
~
~                       VIM - Vi IMproved
~
~                        version 7.0.235
~                    by Bram Moolenaar et al.
~           Vim is open source and freely distributable
~
~                   Help poor children in Uganda!
~          type  :help iccf<Enter>       for information
~
~          type  :q<Enter>               to exit
~          type  :help<Enter>  or  <F1>  for on-line help
~          type  :help version7<Enter>   for version info
~
~
~
~
~
                                                0,0-1         All
```

Vim est un programme un peu surprenant qui ne s'utilise pas comme la plupart des éditeurs de texte que vous connaissez. Il m'a fallu un peu de temps pour m'y habituer et il vous en faudra aussi, mais le jeu en vaut la chandelle.

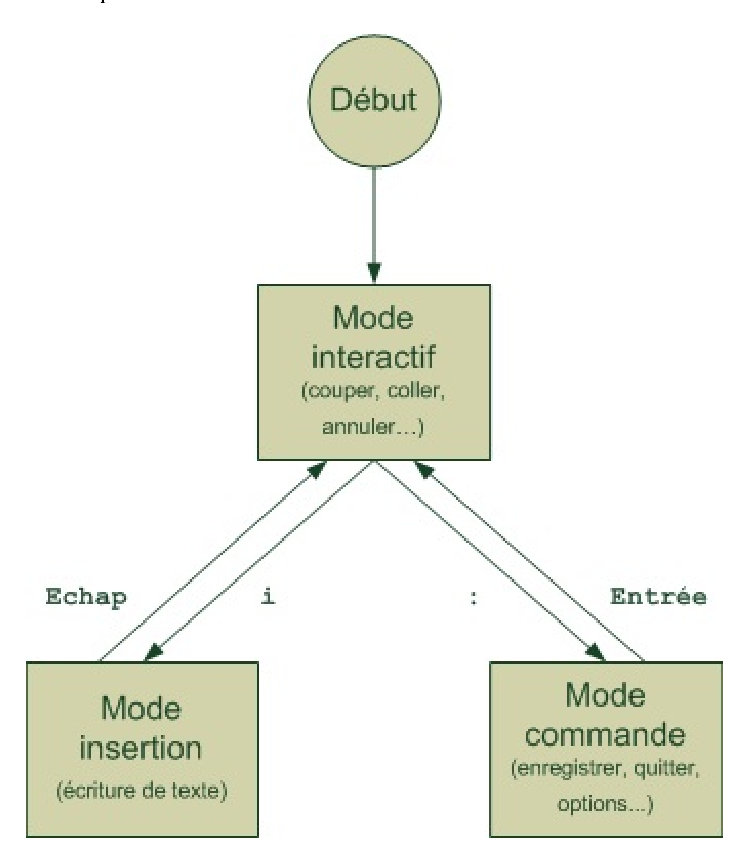
Si on ne vous explique rien, vous risquez d'être un peu perdus. Pire, vous aurez même du mal à sortir de Vim (ne riez pas, ça m'est arrivé la première fois !).

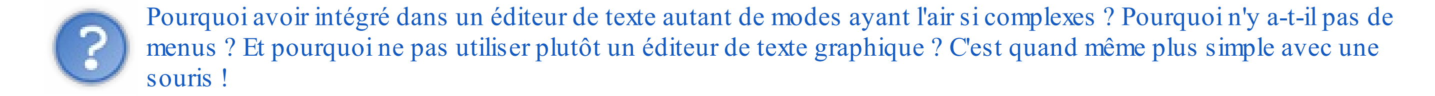
Voilà ce qu'il faut savoir… et qu'on aurait dû m'expliquer dès le départ, d'ailleurs. Vim possède trois modes de travail différents.

- **Mode interactif** : c'est le mode par défaut par lequel vous commencez. En lançant Vim, vous êtes donc en mode interactif. Dans ce mode, vous ne pouvez pas écrire de texte (oui, je sais, il s'agit d'un comble pour un éditeur de texte !). N'essayez donc pas d'appuyer sur des lettres au hasard car vous risqueriez de faire n'importe quoi !Le mode interactif est un mode puissant qui permet de se déplacer dans le texte, de supprimer une ligne, copier-coller du texte, rejoindre une ligne précise, annuler ses actions, etc. Chaque action peut être déclenchée en appuyant sur une touche du clavier (par exemple, on appuie sur `u` pour annuler la dernière action).
- **Mode insertion** : celui-là, c'est celui que vous connaissez ! Vous tapez du texte et ce dernier s'insère à l'endroit où se trouve le curseur.
  Pour entrer dans ce mode, il existe plusieurs possibilités. L'une des plus courantes est d'appuyer sur la touche `i` (*insertion*). Pour en sortir, il faut appuyer sur la touche `Echap`.
- **Mode commande** : ce mode permet de lancer des commandes telles que « quitter », « enregistrer », etc. Vous pouvez aussi l'utiliser pour activer des options de Vim (comme la coloration syntaxique, l'affichage du numéro des lignes…). Vous pouvez même envoyer des commandes au shell (la console) telles que `ls`, `locate`, `cp`, etc.

Pour activer ce mode, vous devez être en mode interactif et appuyer sur la touche `deux points` « : ». Vous validerez la commande avec la touche `Entrée` et reviendrez alors au mode interactif.

Je résume. Vim possède trois modes (figure suivante) : interactif, insertion et commande. Vous démarrez en mode interactif. Le seul mode que vous connaissez et qui ne sera pas nouveau pour vous est le mode insertion. Les deux autres modes (interactif et commande) vont quelque peu vous surprendre.





Pourquoi avoir intégré dans un éditeur de texte autant de modes ayant l'air si complexes ? Pourquoi n'y a-t-il pas de menus ? Et pourquoi ne pas utiliser plutôt un éditeur de texte graphique ? C'est quand même plus simple avec une souris !

Cela fait beaucoup de questions dites donc. 

Je vais essayer de vous répondre simplement et, dans un premier temps, il va falloir que vous me croyiez sur parole : si des gens se sont amusés à créer tous ces « modes » et tous ces raccourcis clavier, ce n'est pas juste pour le plaisir tordu de faire des choses compliquées.

En fait, vous allez rapidement vous rendre compte que **vous pouvez faire des choses que vous ne soupçonniez pas réalisables avec un éditeur de texte** : supprimer le mot actuel, couper le texte du curseur jusqu'à la fin de la ligne, coller quatre fois le texte qui se trouve dans le presse-papier, sauter à la ligne n° 453, sauter à la dernière ligne, etc.

Toutes ces choses-là se font au clavier et, pour la plupart d'entre elles, vous devrez retenir par cœur quelle touche correspond à quelle action. C'est un peu contraignant au départ, mais imaginez que c'est comme apprendre à taper des dix doigts au clavier comme un dactylo : au début, c'est difficile ; vous avez l'impression de ramer, d'aller moins vite qu'avant, mais petit à petit vous gagnez en productivité, vous allez de plus en plus vite et vous finissez par vous demander comment vous avez pu rester autant de temps sans connaître tout ça.



Et pour ceux qui voudraient une interface graphique, sachez que Vim a été porté en interface graphique sous le nom « gVim » (ou vim-gnome selon les versions). Vous pouvez donc l'installer (même si vous utilisez KDE, cela fonctionnera) et le lancer : le fonctionnement est identique à celui du Vim de la console. Il est même disponible en version Windows (figure suivante)… si ce n'est pas beau, ça !
Par défaut, cette fenêtre affiche des menus et une barre d'outils, comme un éditeur de texte classique. Un habitué du Vim console aura bien entendu plutôt tendance à utiliser les raccourcis clavier, qui permettent de gagner du temps.

Opérations basiques (déplacement, écriture, enregistrement…)

Nous allons découvrir Vim au travers de plusieurs étapes. Celles-ci deviendront de plus en plus complexes, mais nous allons commencer en douceur. ;)

L'ouverture de Vim

Pour le moment, si vous lancez Vim en tapant simplement la commande suivante sans aucun paramètre :

Code : Console

```
vim
```

… il s'ouvre sur un nouveau fichier vide que vous avez déjà vu (figure suivante).

```
~
~
~
~
~
~                      VIM - Vi IMproved
~
~                       version 7.0.235
~                   by Bram Moolenaar et al.
~           Vim is open source and freely distributable
~
~                  Help poor children in Uganda!
~          type  :help iccf<Enter>       for information
~
~          type  :q<Enter>               to exit
~          type  :help<Enter>  or  <F1>  for on-line help
~          type  :help version7<Enter>   for version info
~
~
~
~
~
                                              0,0-1         All
```

Vous pouvez aussi ouvrir un fichier en ajoutant son nom en paramètre :

Code : Console

```
vim nomdufichier
```

Si le fichier n'existe pas, il sera créé.

`i` : insérer du texte

Nous allons partir d'un fichier vide. Nous souhaitons commencer par entrer du texte (quoi de plus normal pour un éditeur de texte, après tout ?).

Appuyez sur `i` (« i » minuscule). Vous basculez alors en mode insertion ; à présent, il vous est possible de taper du texte (figure suivante).

```
Salut les Zéros !

J'écris juste quelques lignes dans Vim pour remplir l'écran.
Mine de rien, une fois qu'on a basculé en mode insertion, écrire du texte n'est
pas compliqué. ;o)
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                  4,102-99      All
```

Notez le message `-- INSERT --` en bas de l'écran, qui vous confirme que vous êtes en mode insertion.

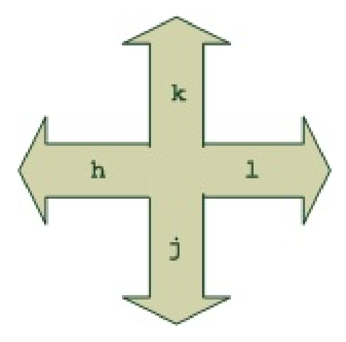
Écrivez quelques lignes comme moi puis appuyez sur la touche `Echap` pour revenir au mode interactif (le mode normal dans lequel vous vous trouviez au départ). Le message `-- INSERT --` disparaît alors et vous revoilà en mode interactif.

Le déplacement

`h`, `j`, `k`, `l` : se déplacer dans tous les sens

En mode interactif, il est possible de déplacer le curseur au sein du texte. Pour cela, on utilise les touches :

- `h` : aller à gauche ;
- `j` : aller en bas ;
- `k` : aller en haut ;
- `l` : aller à droite.




QUOIII ? C'est le comble ! On ne peut même pas utiliser les flèches du clavier pour se déplacer ?!

Si si, vous pouvez également les utiliser : vous n'avez qu'à essayer pour voir. D'ailleurs, en mode insertion, c'est la seule chose qui fonctionne.

0 et $: se déplacer en début et fin de ligne

Pour placer le curseur au tout début de la ligne, appuyez sur `0` en mode interactif.
La touche `Origine` que vous avez peut-être l'habitude d'utiliser fonctionne aussi. Cependant, retenez plutôt qu'il faut utiliser `0`, ça vous sera utile par la suite.

De même, pour se rendre en fin de ligne, appuyez sur la touche `$`.
Là encore, la touche `Fin` fonctionne elle aussi, mais essayez de prendre l'habitude d'utiliser `$` ; ce sera payant, vous allez voir.

w : se déplacer de mot en mot

Avec `w`, vous pouvez vous déplacer de mot en mot dans le fichier. C'est un autre moyen, parfois plus efficace et plus rapide, pour se déplacer au sein d'une ligne du fichier.

:w : enregistrer le fichier

Pour enregistrer votre fichier, vous devez être au préalable en mode interactif (appuyez sur `Echap` pour vous en assurer).

Appuyez ensuite sur la touche `deux points` « : » pour passer en mode commande, puis tapez `w` (*write*) suivi du nom du fichier. La commande doit s'afficher en bas.

Dans mon cas, j'ai donc tapé `:w monfichier` (figure suivante). Appuyez ensuite sur la touche `Entrée` pour valider. Le bas de l'écran doit indiquer que le fichier a été écrit (*written*) :

Code : Console

```
"monfichier" [New] 4L, 185C written                        4,101-
98      All
```



Notez que j'aurais tout aussi bien pu donner une extension `.txt` à mon fichier.

```
Salut les Zéros !

J'écris juste quelques lignes dans Vim pour remplir l'écran.
Mine de rien, une fois qu'on a basculé en mode insertion, écrire du texte n'est
pas compliqué. ;o)
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:w monfichier
```

:q : quitter

Maintenant que vous avez enregistré, vous pouvez quitter Vim en tapant `:q`.



J'ai essayé de quitter en ayant fait des modifications après avoir enregistré et un message d'erreur s'affiche en rouge : « *No write since last change* ».

Vim vous interdit de quitter si vous n'avez pas enregistré vos changements. Vous pouvez toutefois forcer la fermeture du logiciel en ajoutant un point d'exclamation à la fin : `:q!`. Cette fois, il n'y aura aucune erreur.

:wq : enregistrer puis quitter

C'est la combinaison des deux commandes que nous venons de voir. Vous enregistrez et quittez immédiatement Vim lorsque vous tapez `:wq`.

Opérations standard (copier, coller, annuler…)

Nous avons vu le strict minimum de ce qu'il faut connaître pour se débrouiller dans Vim. Si cela n'a rien de difficile, il faut bien avouer que c'est tout de même perturbant. Prenez donc le temps de vous y habituer.

À présent, allons un peu plus loin. Vous allez d'ailleurs commencer à trouver Vim pratique (et parfois même étonnant). Nous allons effectuer la majorité de ces actions en mode interactif : appuyez sur la touche `Echap` si vous n'y êtes pas déjà.

x : effacer des lettres

Placez le curseur sur une lettre en mode interactif puis appuyez sur `x` pour l'effacer.
Cela revient à appuyer sur `Suppr` en mode insertion.

On peut aller plus loin et effacer plusieurs lettres d'un coup. Pour cela, utilisez la formule suivante :

`(nombre)x`

Par exemple, si vous tapez `4x` (4 puis x), vous supprimerez les quatre prochaines lettres en partant du curseur.



Vous devez taper `4` puis `x`. Ne vous étonnez pas si rien ne s'affiche à l'écran lorsque vous tapez `4` : c'est normal. Écrivez la commande jusqu'au bout, cela fonctionnera.

d : effacer des mots, des lignes…

De la même manière, on utilise la touche `d` pour supprimer des mots et des lignes.

Commençons par supprimer une ou plusieurs lignes.

dd : supprimer une ligne

Appuyez deux fois sur `d` (`dd`) pour supprimer toute la ligne sur laquelle se trouve le curseur.

Mieux : vous pouvez faire précéder cette instruction d'un nombre de lignes à supprimer. Par exemple, si vous tapez `2dd`, vous supprimerez deux lignes d'un coup.



Encore une fois, ne vous étonnez pas si juste après avoir tapé `2` rien ne s'affiche à l'écran. L'information est gardée en mémoire par Vim, mais l'action ne sera vraiment exécutée que lorsque vous aurez tapé entièrement `2dd`.

Note importante : la ligne ainsi supprimée est en fait « coupée » et placée en mémoire. Elle peut être collée, comme on le verra plus loin, avec la touche `p`.

dw : supprimer un mot

Placez le curseur sur la première lettre d'un mot. Tapez ensuite `dw` (*delete word*) : cela supprime le mot complet !
Si le curseur est positionné au milieu du mot, vous ne supprimerez que les prochains caractères de celui-ci (jusqu'à l'espace qui suit).

Vous pouvez aussi supprimer les trois prochains mots en tapant `3dw`. Notez que le `3` peut être placé entre le `d` et le `w` ; cela revient au même : `d3w` (qui peut se lire « *delete 3 words* »).

d0 et d$: supprimer le début ou la fin de la ligne

Vous souvenez-vous de `0` et de `$` ? Je vous avais demandé de les utiliser à la place des touches `Origine` et `Fin` car nous en aurions à nouveau besoin par la suite. Le moment est venu de s'en resservir.

- En tapant `d0`, vous supprimez du curseur jusqu'au début de la ligne.
- En tapant `d$`, vous supprimez du curseur jusqu'à la fin de la ligne.

Pratique !

yy : copier une ligne en mémoire

`yy` copie la ligne actuelle en mémoire.
Cela fonctionne comme `dd`, qui lui la « coupe ». Vous pouvez aussi utiliser `yw` pour copier un mot, `y$` pour copier du curseur jusqu'à la fin de la ligne, etc.

p : coller

Si vous avez « coupé » du texte avec `dd` ou copié du texte avec `yy` (ou un de leurs équivalents) vous pouvez ensuite le coller avec la touche `p`.



Attention, retenez bien ceci : si vous avez copié une ligne en mémoire et que vous appuyez sur `p`, elle sera collée sur **la ligne située après le curseur**.

 On est parfois surpris de voir où se colle le texte ; prenez donc le temps de vous y habituer.

Vous pouvez aussi coller plusieurs fois un texte en faisant précéder le `p` d'un nombre. Par exemple, `8p` collera huit fois le texte en mémoire.

Si je place mon curseur sur une ligne, que je tape `yy` puis `8p`, je la collerai donc huit fois (figure suivante) !

```
Salut les Zéros !

J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
Mine de rien, une fois qu'on a basculé en mode insertion, écrire du texte n'est
pas compliqué. ;o)
~
~
~
~
~
~
~
~
~
~
8 more lines                                                  4,1          All
```

`r` : remplacer une lettre

Si vous avez fait une faute sur une lettre seulement, vous pouvez passer en mode remplacement.

Placez le curseur sur la lettre à remplacer. Tapez « r » suivi de la lettre que vous voulez mettre à la place. Par exemple, `rs` remplace la lettre actuelle par un « s ».

Si vous utilisez un « R » majuscule, vous basculerez cette fois dans le mode remplacement : vous pourrez alors remplacer plusieurs lettres à la fois. Vous pouvez par exemple écrire « Rbonjour » pour remplacer les caractères par « bonjour ».
Pour revenir au mode interactif normal, appuyez sur `Echap`.

`u` : annuler les modifications

Pour annuler vos dernière modifications, appuyez sur `u` (*undo*). Si vous souhaitez annuler vos quatre dernières modifications, appuyez sur `4u`.
Vous commencez à connaître la formule, c'est toujours la même. :D

Pour répéter un changement (= annuler une annulation), appuyez sur `Ctrl + R`.

`G` : sauter à la ligne n° X

Toutes les lignes d'un fichier possèdent un numéro. La numérotation commence à 1.

Regardez bien en bas à droite de Vim, vous devriez voir quelque chose comme 4,3.
4 correspond au numéro de la ligne sur laquelle se trouve le curseur, et 3 au numéro de la colonne (3e lettre de la ligne).

Vous pouvez par exemple directement sauter à la ligne n° 7 en tapant `7G` (attention, c'est un « G » majuscule, donc pensez à laisser la touche `Maj` appuyée).

Pour sauter à la dernière ligne, tapez simplement `G`.
Pour revenir à la première ligne, tapez `gg`.

Opérations avancées (split, fusion, recherche…)

Nous avons vu l'essentiel des commandes les plus courantes. Nous allons maintenant découvrir une série de commandes un peu plus complexes parmi lesquelles la fusion de fichiers, la recherche, le remplacement, le découpage de l'écran (*split*), etc.

Toutes ces commandes se lancent depuis le mode interactif.

/ : rechercher un mot

Si vous tapez `/`, vous passez en mode recherche. Le curseur se place en bas de l'écran (vous indiquant que vous êtes passés en mode commande).
Écrivez ensuite le mot que vous recherchez, par exemple « remplir » : `/remplir`. Tapez ensuite sur `Entrée` pour valider.

Le curseur se place alors sur la prochaine occurrence de « remplir » dans le fichier.
Pour passer à la prochaine occurrence du mot, plus bas dans le fichier (s'il apparaît plusieurs fois), appuyez sur `n`. Pour rechercher en arrière, appuyez sur `N` (`Maj + n`).



Si vous souhaitez dès le départ lancer une recherche qui remonte vers le début du fichier, utilisez `?` au lieu de `/` pour lancer la recherche ; le fonctionnement reste le même.

`:s` : rechercher et remplacer du texte

Pour rechercher et remplacer du texte, c'est un peu plus compliqué. Il y a en effet plusieurs façons d'effectuer le remplacement.

La plus simple façon d'effectuer une recherche consiste à taper `:s/ancien/nouveau` pour rechercher « ancien » et le remplacer par « nouveau ». Le problème… c'est que cela ne remplacera que la première occurrence d'« ancien » par « nouveau ».

Voici toutes les variantes à connaître :

- `:s/ancien/nouveau` : remplace la première occurrence de la ligne où se trouve le curseur ;
- `:s/ancien/nouveau/g` : remplace toutes les occurrences de la ligne où se trouve le curseur ;
- `:#,#s/ancien/nouveau/g` : remplace toutes les occurrences dans les lignes n° # à # du fichier ;
- `:%s/ancien/nouveau/g` : remplace toutes les occurrences dans tout le fichier. C'est peut-être ce que vous utiliserez le plus fréquemment.

`:r` : fusion de fichiers

Avec `:r`, vous pouvez insérer un fichier à la position du curseur. Vous devez indiquer le nom du fichier à insérer, par exemple : `:r autrefichier`.

L'autocomplétion avec `Tab` fonctionne là aussi, donc pas besoin d'écrire le nom du fichier en entier !

Le découpage d'écran (split)

Vim possède une fonctionnalité pratique : il permet de découper l'écran et d'ouvrir plusieurs fichiers.

`:sp` : découper l'écran horizontalement

Le plus simple pour commencer est de découper l'écran horizontalement. Tapez la commande `:sp` pour scinder l'écran en deux, comme sur la figure suivante.

```
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
@
monfichier [+]                                                3,1            66%

J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
J'écris juste quelques lignes dans Vim pour remplir l'écran.
monfichier [+]                                                3,1            33%
:sp
```

Le fichier est ouvert une seconde fois (ce qui vous permet de voir deux endroits différents du fichier à la fois) mais il est bien entendu possible d'ouvrir deux fichiers différents. Pour cela, ajoutez le nom du fichier à ouvrir à la suite de la commande : `:sp autrefichier`. Bonne nouvelle : l'autocomplétion à l'aide de la touche `Tab` fonctionne aussi dans Vim !

Vous pouvez cette fois-ci taper à nouveau `:sp` pour scinder l'écran en trois et ainsi de suite, mais gare à la lisibilité !

:vsp : découper l'écran verticalement

Si le découpage horizontal par défaut ne vous convient pas, sachez que vous pouvez aussi effectuer un découpage vertical avec `:vsp` (figure suivante).

```
Salut les Zéros !                        |Salut les Zéros !
                                         |
J'écris juste quelques lignes dans Vim p|J'écris juste quelques lignes dans Vim
our remplir l'écran.                     |pour remplir l'écran.
J'écris juste quelques lignes dans Vim p|J'écris juste quelques lignes dans Vim
our remplir l'écran.                     |pour remplir l'écran.
J'écris juste quelques lignes dans Vim p|J'écris juste quelques lignes dans Vim
our remplir l'écran.                     |pour remplir l'écran.
J'écris juste quelques lignes dans Vim p|J'écris juste quelques lignes dans Vim
our remplir l'écran.                     |pour remplir l'écran.
J'écris juste quelques lignes dans Vim p|J'écris juste quelques lignes dans Vim
our remplir l'écran.                     |pour remplir l'écran.
J'écris juste quelques lignes dans Vim p|J'écris juste quelques lignes dans Vim
our remplir l'écran.                     |pour remplir l'écran.
J'écris juste quelques lignes dans Vim p|J'écris juste quelques lignes dans Vim
our remplir l'écran.                     |pour remplir l'écran.
J'écris juste quelques lignes dans Vim p|J'écris juste quelques lignes dans Vim
our remplir l'écran.                     |pour remplir l'écran.
J'écris juste quelques lignes dans Vim p|J'écris juste quelques lignes dans Vim
our remplir l'écran.                     |pour remplir l'écran.
J'écris juste quelques lignes dans Vim p|J'écris juste quelques lignes dans Vim
our remplir l'écran.                     |pour remplir l'écran.
monfichier [+]        3,1          Top monfichier [+]        3,1           Top
:vsp
```

Il est bien entendu possible de répéter plusieurs fois la commande et même de combiner des découpages verticaux et horizontaux.

Les principaux raccourcis en écran splitté

Chaque morceau de l'écran (correspondant à un fichier) est appelé **viewport**.
Voici une liste de raccourcis pratiques que vous pouvez utiliser lorsque l'écran est splitté (scindé).

- `Ctrl + w` puis `Ctrl + w` : navigue de viewport en viewport. Répétez l'opération plusieurs fois pour accéder au viewport désiré.
- `Ctrl + w` puis `j` : déplace le curseur pour aller au viewport juste en dessous. La même chose fonctionne avec les touches `h`, `k` et `l` que l'on utilise traditionnellement pour se déplacer dans Vim.
- `Ctrl + w` puis `+` : agrandit le viewport actuel.
- `Ctrl + w` puis `-` : réduit le viewport actuel.
- `Ctrl + w` puis `=` : égalise à nouveau la taille des viewports.
- `Ctrl + w` puis `r` : échange la position des viewports. Fonctionne aussi avec « R » majuscule pour échanger en sens inverse.
- `Ctrl + w` puis `q` : ferme le viewport actuel.

Voilà qui devrait vous permettre de faire ce que vous voulez en écran splitté.

: ! : lancer une commande externe

Il est possible d'écrire des commandes traditionnelles du shell directement dans Vim. Pour cela, commencez par taper `:!` suivi du nom de la commande.

Essayez par exemple de taper `:!ls`. Vous afficherez alors le contenu du dossier dans lequel vous vous trouvez !
Cette fonctionnalité est bien pratique pour effectuer quelques actions sans avoir à quitter Vim.

Les options de Vim

Vim peut être personnalisé de deux façons différentes :

- En activant ou désactivant des options. La documentation complète des options est disponible en ligne.
- En installant des plugins. Voyez la page officielle des plugins les plus téléchargés de Vim.

Nous n'allons pas passer en revue les plugins, mais il y a un certain nombre d'options intéressantes qui valent le coup d'être activées.

Le fonctionnement des options

Les options peuvent être activées après le démarrage de Vim en lançant des commandes. Cependant, ces options seront « oubliées » dès que vous quitterez le logiciel.
Si vous voulez que les options soient activées à chaque démarrage de Vim, il faut créer un fichier de configuration `.vimrc` dans votre répertoire personnel.

Activer des options en mode commande

La première méthode consiste à activer l'option en mode commande. Une fois Vim ouvert, pour activer l'option nommée « option », tapez :

```
:set option
```

Pour la désactiver, tapez :

```
:set nooption
```

Il faut donc ajouter le préfixe `no` devant le nom de l'option pour la désactiver.

Certaines options doivent être précisées avec une valeur, comme ceci :

`:set option=valeur`

Pour connaître l'état d'une option :

`:set option?`

Activer des options dans un fichier de configuration

C'est à mon avis la meilleure façon de procéder. Commencez par copier un fichier de configuration déjà commenté qui vous servira d'exemple : il y en a un dans `/etc/vim` qui s'appelle `vimrc`.

Copiez-le dans votre répertoire personnel en le faisant précéder d'un point (pour que ce soit un fichier caché) :

Code : Console

```
$ cp /etc/vim/vimrc  ~/.vimrc
```

Ouvrez maintenant ce fichier… avec Vim, bien sûr.

Code : Console

```
$ vim .vimrc
```

Le début du fichier ressemble à ceci :

Code : Console

```
" All system-wide defaults are set in $VIMRUNTIME/debian.vim (usually just
" /usr/share/vim/vimcurrent/debian.vim) and sourced by the call to :runtime
" you can find below.  If you wish to change any of those settings, you should
" do it in this file (/etc/vim/vimrc), since debian.vim will be overwritten
" everytime an upgrade of the vim packages is performed.  It is recommended to
" make changes after sourcing debian.vim since it alters the value of the
" 'compatible' option.

" This line should not be removed as it ensures that various options are
" properly set to work with the Vim-related packages available in Debian.
runtime! debian.vim

" Uncomment the next line to make Vim more Vi-compatible
" NOTE: debian.vim sets 'nocompatible'.  Setting 'compatible' changes numerous
" options, so any other options should be set AFTER setting 'compatible'.
"set compatible

" Vim5 and later versions support syntax highlighting. Uncommenting the next
" line enables syntax highlighting by default.
"syntax on

" If using a dark background within the editing area and syntax highlighting
" turn on this option as well
```

Les lignes commençant par « " » sont des commentaires. Je vous recommande de les lire, ils fournissent des informations utiles.

Passons maintenant à l'activation de quelques commandes bien utiles. Je vous recommande de travailler comme moi, avec le fichier de configuration `.vimrc`, et d'activer les options qui vous plaisent en décommentant les lignes concernées.

Pour cela, la meilleure façon de procéder est de se mettre en mode interactif, de se déplacer avec `hjkl` et d'appuyer sur `x` lorsque le curseur est sur un guillemet pour le supprimer et activer ainsi l'option.

`syntax` : activer la coloration syntaxique

Il s'agit clairement de la première option à activer : la coloration syntaxique. En fonction du type de fichier que vous ouvrez, Vim colorera le texte.

Vim supporte un très très grand nombre de langages de programmation : C, C++, Python, Java, Ruby, Bash, Perl, etc.

Activez donc l'option :

Code : Console

```
syntax on
```



Notez qu'il faut enregistrer, quitter et relancer Vim pour que le changement soit pris en compte… sauf bien sûr si vous activez l'option à la volée en tapant dans Vim : `set syntax=ON`.

`background` : coloration sur un fond sombre

Par défaut, la coloration de Vim est plus adaptée aux fonds clairs. Les commentaires, par exemple, sont écrits en bleu foncé sur noir… ce qui n'est pas très lisible.

Si votre console est sur fond noir (comme chez moi), je vous recommande d'activer la prochaine option `background` et de la mettre à `dark`.

Code : Console

```
set background=dark
```

Les couleurs seront largement plus adaptées.

`number` : afficher les numéros de ligne

Il est possible d'afficher le numéro de chaque ligne à gauche (figure suivante) :

Code : Console

```
set number
```

Cela s'avère assez pratique, notamment quand on programme.

```
  1 Salut les Zéros !
  2
  3 J'écris juste quelques lignes dans Vim pour remplir l'écran.
  4 J'écris juste quelques lignes dans Vim pour remplir l'écran.
  5 J'écris juste quelques lignes dans Vim pour remplir l'écran.
  6 J'écris juste quelques lignes dans Vim pour remplir l'écran.
  7 J'écris juste quelques lignes dans Vim pour remplir l'écran.
  8 J'écris juste quelques lignes dans Vim pour remplir l'écran.
  9 J'écris juste quelques lignes dans Vim pour remplir l'écran.
 10 J'écris juste quelques lignes dans Vim pour remplir l'écran.
 11 J'écris juste quelques lignes dans Vim pour remplir l'écran.
 12 J'écris juste quelques lignes dans Vim pour remplir l'écran.
 13 Mine de rien, une fois qu'on a basculé en mode insertion, écrire du texte n'
    est pas compliqué. ;o)
~
~
~
~
~
~
~
~
~
:set number                                                   6,1           Tout
```

showcmd : afficher la commande en cours

Lorsque vous écrivez une commande comme 2dd pour supprimer deux lignes, vous écrivez à l'aveugle. Vous ne voyez pas ce que vous avez écrit.

Contrairement à Vi, Vim permet de pallier ce problème… encore faut-il activer l'option :

Code : Console

```
set showcmd
```

ignorecase : ignorer la casse lors de la recherche

Lors d'une recherche, si vous souhaitez que Vim ne fasse pas la différence entre les majuscules et les minuscules, activez cette option :

Code : Console

```
set ignorecase
```

mouse : activer le support de la souris

Eh oui ! Même en mode console, il est possible d'utiliser la souris.
Commencez par activer le support de cette dernière :

Code : Console

```
set mouse=a
```

Désormais, vous pourrez cliquer avec la souris sur une lettre pour y déplacer le curseur directement. Vous pourrez également utiliser la molette de la souris pour vous déplacer dans le fichier.



Il vous sera également possible de sélectionner du texte à l'aide de la souris. Vous passerez alors en mode visuel. Dans ce mode, vous pouvez supprimer le texte sélectionné (avec `x`, comme d'habitude), mais aussi mettre le texte tout en majuscules (`U`), minuscules (`u`), etc.

En résumé

- Vim est un éditeur de texte très puissant en console et qui offre plus de possibilités que Nano, que nous avons découvert plus tôt dans cet ouvrage. Son grand concurrent est Emacs.
- Dans Vim, il existe trois modes : interactif, insertion et commande.
- Le mode par défaut est le mode interactif. Il faut appuyer sur la touche `i` pour insérer du texte et sur la touche `Echap` pour revenir au mode interactif.
- On peut lancer des commandes en appuyant sur la touche `deux points` « : » depuis le mode interactif. Par exemple, `:w` enregistre le fichier, `:q` quitte Vim et `:wq` effectue les deux à la fois.
- Il existe de nombreux raccourcis à connaître pour bien utiliser Vim ; il faut prendre le temps de les apprendre pour exploiter pleinement le logiciel.
- On peut modifier le fichier `.vimrc` pour activer certaines options de Vim, comme la coloration automatique du code.

 Introduction aux scripts shell

Vous venez d'apprendre à utiliser un éditeur de texte puissant comme Vim. Cela va vous être particulièrement utile pour les chapitres à venir.

Entrons maintenant dans le vif du sujet : **la programmation shell**. De quoi s'agit-il ?
Imaginez un minilangage de programmation intégré à Linux. Ce n'est pas un langage aussi complet que peuvent l'être le C, le C++ ou le Java par exemple, mais cela permet d'automatiser la plupart de vos tâches : sauvegarde des données, surveillance de la charge de votre machine, etc.

On aurait très bien pu faire tout cela en créant un programme en C par exemple. Le gros avantage du langage shell est d'être totalement intégré à Linux : il n'y a rien à installer, rien à compiler. Et surtout : vous avez très peu de nouvelles choses à apprendre. En effet, toutes les commandes que l'on utilise dans les scripts shell sont des commandes du système que vous connaissez déjà : `ls`, `cut`, `grep`, `sort`…

On parlera beaucoup de shell dans cette section. De quoi s'agit-il exactement ? Nous répondrons à cette question en premier. Ensuite, nous réaliserons notre tout premier script shell qui affiche un message à l'écran… et nous pourrons alors passer aux choses sérieuses dès le chapitre suivant !

Qu'est-ce qu'un shell ?

Dès le début de ce livre, j'ai fait la distinction entre les deux environnements très différents disponibles sous Linux :

- l'environnement console ;
- l'environnement graphique.

La plupart du temps, sur sa machine, on a tendance à utiliser l'environnement graphique, qui est plus intuitif. Cependant, la console est aussi un allié très puissant qui permet d'effectuer des actions habituellement difficiles à réaliser dans un environnement graphique.

Je vous avais dit qu'il y avait plusieurs environnements graphiques disponibles (Unity, KDE, XFCE…) mais qu'il n'y avait qu'une seule console. J'ai menti.

Il existe plusieurs environnements console : les shells

La différence est moins tape-à-l'œil que dans le mode graphique (où l'on voit tout de suite que les menus ne sont pas à la même place, par exemple).
La console a toujours un fond noir et un texte blanc, je vous rassure (quoique ça se personnalise, ça). En revanche, les fonctionnalités offertes par l'invite de commandes peuvent varier en fonction du **shell** que l'on utilise.

 Les différents environnements console sont appelés des shells, c'est ça ?

C'est ça, en effet. Voici les noms de quelques-uns des principaux shells qui existent.

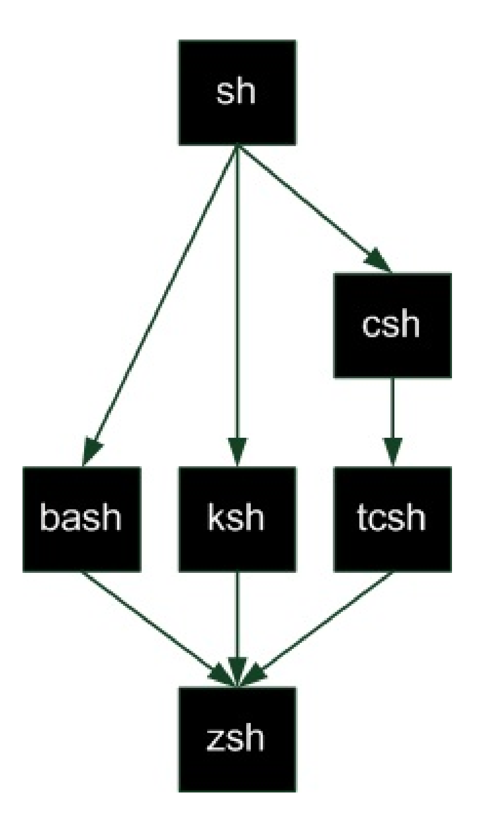
- **sh** : *Bourne Shell*. L'ancêtre de tous les shells.
- **bash** : *Bourne Again Shell*. Une amélioration du *Bourne Shell*, disponible par défaut sous Linux et Mac OS X.
- **ksh** : *Korn Shell*. Un shell puissant assez présent sur les Unix propriétaires, mais aussi disponible en version libre, compatible avec bash.
- **csh** : *C Shell*. Un shell utilisant une syntaxe proche du langage C.
- **tcsh** : *Tenex C Shell*. Amélioration du *C Shell*.
- **zsh** : *Z Shell*. Shell assez récent reprenant les meilleures idées de bash, ksh et tcsh.

Il y en a quelques autres, mais vous avez là les principaux.

Que faut-il savoir ? Tout d'abord que l'ancêtre de tous les shells est le sh (*Bourne Shell*). C'est le plus vieux et il est installé sur tous les OS basés sur Unix. Il est néanmoins pauvre en fonctionnalités par rapport aux autres shells.

Le bash (*Bourne Again Shell*) est le shell par défaut de la plupart des distributions Linux mais aussi celui du terminal de Mac OS X. Il y a fort à parier que c'est celui que vous utilisez en ce moment sous Linux.
Le bash est une amélioration du sh.

Voici dans les grandes lignes comment ont évolué les shells. Chacun hérite de la plupart des fonctionnalités de son ancêtre (figure suivante).



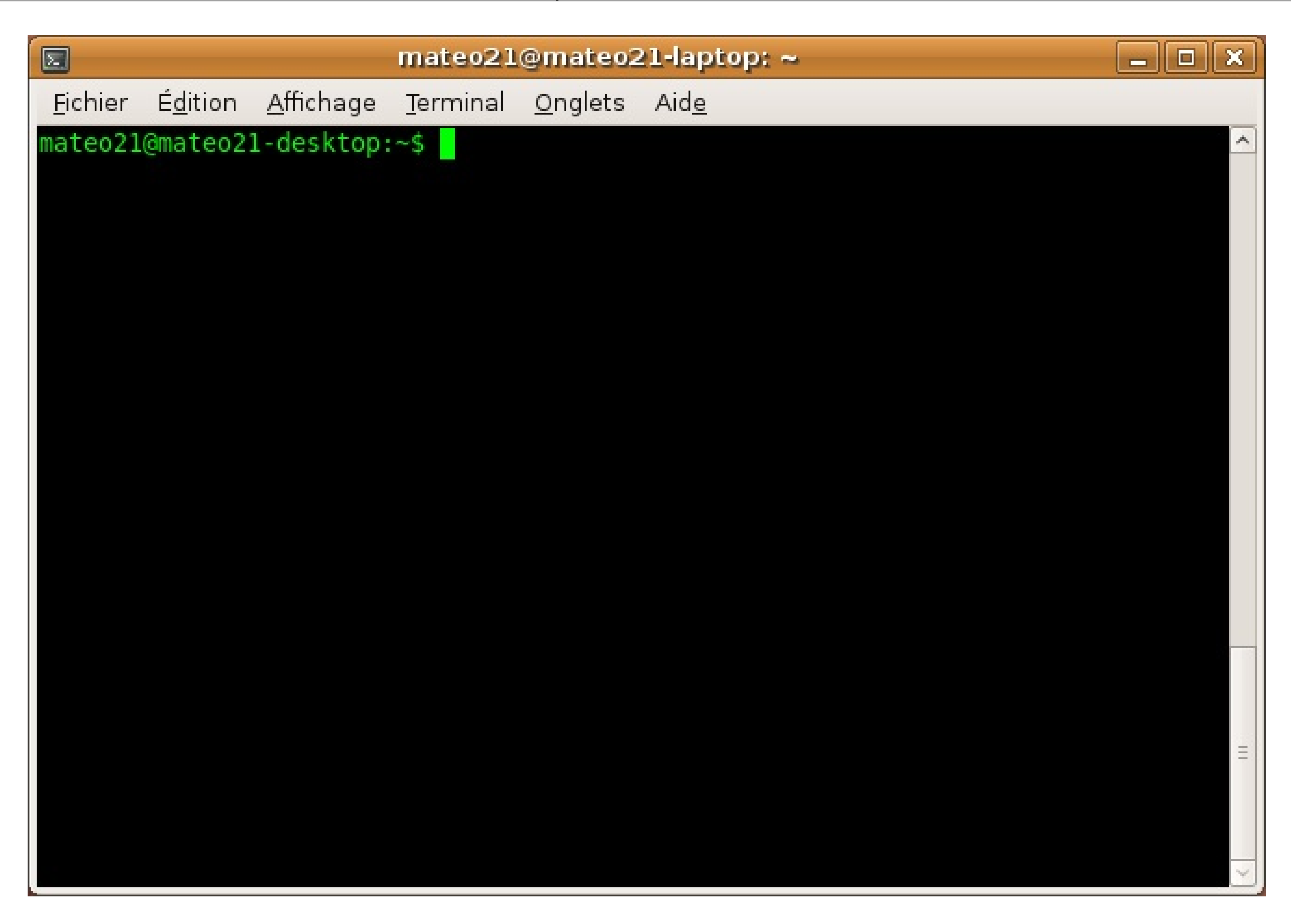


À quoi peut bien servir le sh aujourd'hui alors, si bash est par défaut sous Linux ?

sh reste toujours plus répandu que bash. En fait, vous pouvez être sûrs que tous les OS basés sur Unix possèdent sh, mais ils n'ont pas tous forcément bash. Certains OS basés sur Unix, notamment les OS propriétaires (AIX et Solaris…), utilisent d'autres types de shells ; le ksh y est par exemple très répandu.

À quoi sert un shell ?

Le shell est le programme qui gère l'invite de commandes. C'est donc le programme qui attend que vous rentriez des commandes (comme l'illustre la figure suivante).



C'est aussi le programme qui est capable par exemple de :

- se souvenir quelles étaient les dernières commandes tapées (vous remontez dans votre historique en appuyant sur la flèche « Haut » ou en faisant une recherche avec un `Ctrl + R`) ;
- autocompléter une commande ou un nom de fichier lorsque vous appuyez sur `Tab` (figure suivante) ;

```
mateo21@mateo21-laptop: ~
Fichier  Édition  Affichage  Terminal  Onglets  Aide
mateo21@mateo21-desktop:~$ cu
cupsaccept       cupsd          cups-genppd.5.2    cupstestdsc
cupsaddsmb       cupsdisable    cups-genppdupdate  cupstestppd
cups-calibrate   cupsenable     cupsprofile        cut
cupsctl          cupsfilter     cupsreject
mateo21@mateo21-desktop:~$ cu
```

- gérer les processus (envoi en arrière-plan, mise en pause avec `Ctrl + Z`...) ;
- rediriger et chaîner les commandes (les fameux symboles >, <, |, etc.) ;
- définir des alias (par exemple `ll` signifie chez moi `ls -lArth`).

Bref, le shell fournit toutes les fonctionnalités de base pour pouvoir lancer des commandes.

Souvenez-vous : nous avions modifié un fichier `.bashrc` dans un des premiers chapitres (celui où nous avons appris à utiliser Nano). Le `.bashrc` est le fichier de configuration du bash que Linux vous fait utiliser par défaut. Chaque personne peut avoir son `.bashrc` pour personnaliser son invite de commandes, ses alias, etc.

Installer un nouveau shell

Pour le moment, vous devriez avoir sh et bash installés sur votre système. Si vous voulez essayer un autre shell, comme ksh par exemple, vous pouvez le télécharger comme n'importe quel paquet :

Code : Console

```
# apt-get install ksh
```

Une fois installé, il faut demander à l'utiliser pour votre compte utilisateur. Pour cela, tapez :

Code : Console

```
$ chsh
```

`chsh` signifie *Change Shell*.
On vous demandera où se trouve le programme qui gère le shell. Vous devrez indiquer `/bin/ksh` pour ksh, `/bin/sh` pour sh, `/bin/bash` pour bash, etc.

Quelle importance a tout ceci lorsque l'on réalise un script shell ?

Si je vous parle de cela, c'est parce qu'un script shell **dépend** d'un shell précis. En gros, le langage n'est pas tout à fait le même selon que vous utilisez sh, bash, ksh, etc.

Il est possible d'écrire des scripts sh par exemple. Ceux-là, nous sommes sûrs qu'ils fonctionnent partout car tout le monde possède un shell sh.
Il s'agit toutefois du plus vieux shell, or écrire des scripts en sh est certes possible mais n'est franchement ni facile, ni ergonomique.


Avec quel shell va-t-on écrire nos scripts, alors ?

Je propose d'étudier le bash dans ce cours car :

- on le trouve par défaut sous Linux et Mac OS X (cela couvre assez de monde !) ;
- il rend l'écriture de scripts plus simple que sh ;
- il est plus répandu que ksh et zsh sous Linux.

En clair, le bash est un bon compromis entre sh (le plus compatible) et ksh / zsh (plus puissants).

Notre premier script

Nous allons commencer par écrire un premier script bash tout simple. Il ne sera pas révolutionnaire mais va nous permettre de voir les bases de la création d'un script et comment celui-ci s'exécute. Cela sera donc essentiel pour la suite.

Création du fichier

Commençons par créer un nouveau fichier pour notre script. Le plus simple est d'ouvrir Vim en lui donnant le nom du nouveau fichier à créer :

Code : Console

```
$ vim essai.sh
```

Si `essai.sh` n'existe pas, il sera créé (ce qui sera le cas ici).


J'ai donné ici l'extension `.sh` à mon fichier. On le fait souvent par convention pour indiquer que c'est un script shell, mais sachez que ce n'est pas une obligation. Certains scripts shell n'ont d'ailleurs pas d'extension du tout.
J'aurais donc pu appeler mon script `essai` tout court.

Indiquer le nom du shell utilisé par le script

Vim est maintenant ouvert et vous avez un fichier vide sous les yeux.

La première chose à faire dans un script shell est d'indiquer… quel shell est utilisé. En effet, comme je vous l'ai dit plus tôt, la syntaxe du langage change un peu selon qu'on utilise sh, bash, ksh, etc.

En ce qui nous concerne, nous souhaitons utiliser la syntaxe de bash, plus répandu sous Linux et plus complet que sh. Nous indiquons où se trouve le programme bash :

Code : Console

```
#!/bin/bash
```



Le #! est appelé le **sha-bang**.
`/bin/bash` peut être remplacé par `/bin/sh` si vous souhaitez coder pour sh, `/bin/ksh` pour ksh, etc.

Bien que non indispensable, cette ligne permet de s'assurer que le script est bien exécuté avec le bon shell.
En l'absence de cette ligne, c'est le shell de l'utilisateur qui sera chargé. Cela pose un problème : si votre script est écrit pour bash et que la personne qui l'exécute utilise ksh, il y a de fortes chances pour que le script ne fonctionne pas correctement !

La ligne du sha-bang permet donc de « charger » le bon shell avant l'exécution du script. À partir de maintenant, vous devrez la mettre au tout début de chacun de vos scripts.

Exécution de commandes

Après le sha-bang, nous pouvons commencer à coder.
Le principe est très simple : il vous suffit d'écrire les commandes que vous souhaitez exécuter. Ce sont les mêmes que celles que vous tapiez dans l'invite de commandes !

- `ls` : pour lister les fichiers du répertoire.
- `cd` : pour changer de répertoire.
- `mkdir` : pour créer un répertoire.
- `grep` : pour rechercher un mot.
- `sort` : pour trier des mots.
- etc.

Bref, tout ce que vous avez appris, vous pouvez le réutiliser ici ! ;)

Allez, on va commencer par quelque chose de très simple : un `ls`. On va donc créer un script bash qui va juste se contenter d'afficher le contenu du dossier courant :

Code : Console

```
#!/bin/bash

ls
```

C'est tout !

Les commentaires

Notez que vous pouvez aussi ajouter des commentaires dans votre script. Ce sont des lignes qui ne seront pas exécutées mais qui permettent d'expliquer ce que fait votre script.

Tous les commentaires commencent par un #. Par exemple :

Code : Console

```
#!/bin/bash

# Affichage de la liste des fichiers
ls
```

Vous avez sûrement remarqué que la ligne du sha-bang commence aussi par un #… Oui, c'est un commentaire aussi, mais considérez que c'est un commentaire « spécial » qui a un sens. Il fait un peu exception.

Exécuter le script bash

Nous avons écrit un petit script sans prétention de deux-trois lignes. Notre mission maintenant est de parvenir à l'exécuter.

Commencez par enregistrer votre fichier et fermez votre éditeur. Sous Vim, il suffit de taper `:wq` ou encore `:x`.
Vous retrouvez alors l'invite de commandes.

Donner les droits d'exécution au script

Si vous faites un `ls -l` pour voir votre fichier qui vient d'être créé, vous obtenez ceci :

Code : Console

```
$ ls -l
total 4
-rw-r--r-- 1 mateo21 mateo21 17 2009-03-13 14:33 essai.sh
```

Ce qui nous intéresse ici, ce sont les droits sur le fichier : `-rw-r--r--`.
Si vous vous souvenez un petit peu du chapitre sur les droits, vous devriez vous rendre compte que notre script peut être lu par tout le monde (`r`), écrit uniquement par nous (`w`), et n'est pas exécutable (pas de `x`).

Or, pour exécuter un script, il faut que le fichier ait le droit « exécutable ». Le plus simple pour donner ce droit est d'écrire :

Code : Console

```
$ chmod +x essai.sh
```

Vous pouvez vérifier que le droit a bien été donné :

Code : Console

```
$ ls -l
total 4
-rwxr-xr-x 1 mateo21 mateo21 17 2009-03-13 14:33 essai.sh
```

Tout le monde a maintenant le droit d'exécuter le script. Si vous voulez, vous pouvez limiter ce droit à vous-mêmes mais pour cela je vous invite à revoir le cours sur les droits car je ne vais pas me répéter. :)

Exécution du script

Le script s'exécute maintenant comme n'importe quel programme, en tapant « ./ » devant le nom du script :

Code : Console

```
$ ./essai.sh
essai.sh
```

Que fait le script ? Il fait juste un `ls`, donc il affiche la liste des fichiers présents dans le répertoire (ici, il y avait seulement `essai.sh` dans mon répertoire).

Bien entendu, ce script est inutile ; il était plus simple de taper `ls` directement. Cependant, vous devez vous douter que l'on va pouvoir faire beaucoup mieux que ça dans les prochains chapitres.

Vous pouvez déjà modifier votre script pour qu'avant tout chose il vous donne également le nom du répertoire dans lequel vous vous trouvez :

Code : Console

```
#!/bin/bash

pwd
ls
```

Les commandes seront exécutées une par une :

Code : Console

```
$ ./essai.sh
/home/mateo21/scripts
essai.sh
```

Exécution de débogage

Plus tard, vous ferez probablement de gros scripts et risquerez de rencontrer des bugs. Il faut donc dès à présent que vous sachiez comment déboguer un script.

Il faut l'exécuter comme ceci :

Code : Console

```
$ bash -x essai.sh
```

On appelle en fait directement le programme bash et on lui ajoute en paramètre un `-x` (pour lancer le mode débogage) ainsi que le nom de notre script à déboguer.

Le shell affiche alors le détail de l'exécution de notre script, ce qui peut nous aider à retrouver la cause de nos erreurs :

Code : Console

```
$ bash -x essai.sh
+ pwd
/home/mateo21/scripts
+ ls
essai.sh
```

Créer sa propre commande

Actuellement, le script doit être lancé via `./essai.sh` et vous devez être dans le bon répertoire.
Ou alors vous devez taper le chemin en entier, comme `/home/mateo21/scripts/essai.sh`.


Comment font les autres programmes pour pouvoir être exécutés depuis n'importe quel répertoire sans « ./ » devant ?

Ils sont placés dans un des répertoires du PATH. Le PATH est une variable système qui indique où sont les programmes exécutables sur votre ordinateur. Si vous tapez `echo $PATH` vous aurez la liste de ces répertoires « spéciaux ».

Il vous suffit donc de déplacer ou copier votre script dans un de ces répertoires, comme `/bin`, `/usr/bin` ou `/usr/local/bin` (ou encore un autre répertoire du PATH). Notez qu'il faut être root pour pouvoir faire cela.

Une fois que c'est fait, vous pourrez alors taper simplement `essai.sh` pour exécuter votre programme et ce quel que soit le répertoire dans lequel vous vous trouverez !

Code : Console

```
$ essai.sh
/home/mateo21/scripts
essai.sh
```

En résumé

- Contrairement aux apparences, il existe plusieurs environnements console différents : ce sont les shells. Ce sont eux qui gèrent l'invite de commandes et ses fonctionnalités comme l'historique des commandes, la recherche `Ctrl + R`, l'autocomplétion des commandes…
- Le shell utilisé par défaut sous Ubuntu est bash, mais il existe aussi ksh, zsh, etc.
- Il est possible d'automatiser une série de commandes. On crée pour cela un fichier contenant la liste des commandes à exécuter, appelé *script shell*. On dit que l'on fait de la programmation shell.
- En fonction du shell utilisé, on dispose de différents outils pour créer son script shell. Nous utiliserons ici bash, donc notre fichier de script doit commencer par la ligne `#!/bin/bash`.
- Dans le fichier de script, il suffit d'écrire les commandes à exécuter les unes après les autres, chacune sur une ligne différente.
- Pour exécuter le script (et donc exécuter la liste des commandes qu'il contient) il faut donner les droits d'exécution au fichier (`chmod +x script.sh`) et lancer l'exécution du script avec la commande `./script.sh`.

 Afficher et manipuler des variables

Comme dans tous les langages de programmation, on trouve en bash ce que l'on appelle des **variables**. Elles nous permettent de stocker temporairement des informations en mémoire. C'est en fait la base de la programmation.

Les variables en bash sont assez particulières. Il faut être très rigoureux lorsqu'on les utilise. Si vous avez fait du C ou d'autres langages de programmation, vous allez être un peu surpris par leur mode de fonctionnement ; soyez donc attentifs.
Et si vous n'avez jamais programmé, soyez attentifs aussi. ;)

Déclarer une variable

Nous allons créer un nouveau script que nous appellerons `variables.sh` :

Code : Console

```
$ vim variables.sh
```

La première ligne de tous nos scripts doit indiquer quel shell est utilisé, comme nous l'avons appris plus tôt. Commencez donc par écrire :

Code : Console

```
#!/bin/bash
```

Cela indique que nous allons programmer en bash.

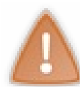
Maintenant, définissons une variable. Toute variable possède un nom et une valeur :

Code : Console

```
message='Bonjour tout le monde'
```

Dans le cas présent :

- la variable a pour **nom** `message` ;
- … et pour **valeur** `Bonjour tout le monde`.



Ne mettez pas d'espaces autour du symbole égal « = » ! Le bash est très pointilleux sur de nombreux points, évitez par conséquent de le vexer.
Je vous signalerai systématiquement les pièges à éviter, car il y en a un certain nombre !



Si vous voulez insérer une apostrophe dans la valeur de la variable, il faut la faire précéder d'un antislash \. En effet, comme les apostrophes servent à délimiter le contenu, on est obligé d'utiliser un **caractère d'échappement** (c'est comme ça que cela s'appelle) pour pouvoir véritablement insérer une apostrophe :

Code : Console

```
message='Bonjour c\'est moi'
```

Bien, reprenons notre script. Il devrait à présent ressembler à ceci :

Code : Console

```
#!/bin/bash

message='Bonjour tout le monde'
```

Exécutez-le pour voir ce qui se passe (après avoir modifié les droits pour le rendre exécutable, bien sûr) :

Code : Console

```
$ ./variables.sh
$
```

Il ne se passe rien !



Que fait le script, alors ?

Il met en mémoire le message `Bonjour tout le monde`, et c'est tout ! Rien ne s'affiche à l'écran !

Pour afficher une variable, il va falloir utiliser une commande dont je ne vous ai pas encore parlé…

echo : afficher une variable

Avant de commencer à parler de **variables**, il y a une commande que j'aimerais vous présenter : `echo`. J'aurais pu en parler avant que l'on commence à faire des scripts bash, mais vous n'en auriez pas vu l'utilité avant d'aborder ce chapitre.

Son principe est très simple : elle affiche dans la console le message demandé. Un exemple :

Code : Console

```
$ echo Salut tout le monde
Salut tout le monde
```

Comme vous le voyez, c'est simple comme bonjour. Les guillemets ne sont pas requis.



Mais… comment est-ce que cela fonctionne ?

En fait, la commande `echo` affiche dans la console tous les paramètres qu'elle reçoit. Ici, nous avons envoyé quatre paramètres :

- `Salut` ;
- `tout` ;
- `le` ;
- `monde`.

Chacun des mots était considéré comme un paramètre que `echo` a affiché. Si vous mettez des guillemets autour de votre message, celui-ci sera considéré comme étant un seul et même paramètre (le résultat sera visuellement le même) :

Code : Console

```
$ echo "Salut tout le monde"
Salut tout le monde
```

Si vous voulez insérer des retours à la ligne, il faudra activer le paramètre `-e` et utiliser le symbole `\n` :

Code : Console

```
$ echo -e "Message\nAutre ligne"
Message
Autre ligne
```

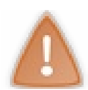
Afficher une variable

Pour afficher une variable, nous allons de nouveau utiliser son nom précédé du symbole dollar `$` :

Code : Console

```
#!/bin/bash

message='Bonjour tout le monde'
echo $message
```



Comparez les lignes 3 et 4 : lorsque l'on **déclare** la variable à la ligne 3, on ne doit pas mettre de `$` devant. En revanche, lorsqu'on l'**affiche** à la ligne 4, on doit cette fois mettre un `$` !

Résultat :

Code : Console

```
Bonjour tout le monde
```

Maintenant, supposons que l'on veuille afficher à la fois du texte et la variable. Nous serions tentés d'écrire :

Code : Console

```
#!/bin/bash

message='Bonjour tout le monde'
echo 'Le message est : $message'
```

Le problème est que cela ne fonctionne pas comme on le souhaite car cela affiche :

Code : Console

```
Le message est : $message
```

Pour bien comprendre ce qui se passe, intéressons-nous au fonctionnement de ce que l'on appelle les « quotes ».

Les quotes

Il est possible d'utiliser des **quotes** pour délimiter un paramètre contenant des espaces. Il existe trois types de quotes :

- les apostrophes ' ' (simples quotes) ;
- les guillemets " " (doubles quotes) ;
- les accents graves ` ` (back quotes), qui s'insèrent avec `Alt Gr + 7` sur un clavier AZERTY français.

Selon le type de quotes que vous utilisez, la réaction de bash ne sera pas la même.

Les simples quotes ' '

Commençons par les simples quotes :

Code : Console

```
message='Bonjour tout le monde'
echo 'Le message est : $message'
```

Code : Console

```
Le message est : $message
```

Avec de simples quotes, la variable n'est pas analysée et le `$` est affiché tel quel.

Les doubles quotes " "

Avec des doubles quotes :

Code : Console

```
message='Bonjour tout le monde'
echo "Le message est : $message"
```

Code : Console

```
Le message est : Bonjour tout le monde
```

… ça fonctionne ! Cette fois, la variable est analysée et son contenu affiché.

En fait, les doubles quotes demandent à bash d'analyser le contenu du message. S'il trouve des symboles spéciaux (comme des variables), il les interprète.
Avec de simples quotes, le contenu était affiché tel quel.

Les back quotes ` `

Un peu particulières, les back quotes demandent à bash d'**exécuter** ce qui se trouve à l'intérieur.

Un exemple valant mieux qu'un long discours, **regardez la première ligne** :

Code : Console

```
message=`pwd`
echo "Vous êtes dans le dossier $message"
```

Code : Console

```
Vous êtes dans le dossier /home/mateo21/bin
```

La commande `pwd` a été exécutée et son contenu inséré dans la variable `message` ! Nous avons ensuite affiché le contenu de la variable.

Cela peut paraître un peu tordu, mais c'est réellement utile. Nous nous en resservirons dans les chapitres suivants.

read : demander une saisie

Vous pouvez demander à l'utilisateur de saisir du texte avec la commande `read`. Ce texte sera immédiatement stocké dans une variable.

La commande `read` propose plusieurs options intéressantes. La façon la plus simple de l'utiliser est d'indiquer le nom de la variable dans laquelle le message saisi sera stocké :

Code : Console

```
read nomvariable
```

Adaptons notre script pour qu'il nous demande notre nom puis qu'il nous l'affiche :

Code : Console

```
#!/bin/bash

read nom
echo "Bonjour $nom !"
```

Lorsque vous lancez ce script, rien ne s'affiche, mais vous pouvez taper du texte (votre nom, par exemple) :

Code : Console

```
Mathieu
Bonjour Mathieu !
```

Notez que la première ligne correspond au texte que j'ai tapé au clavier.

Affecter simultanément une valeur à plusieurs variables

On peut demander de saisir autant de variables d'affilée que l'on souhaite. Voici un exemple de ce qu'il est possible de faire :

Code : Console

```
#!/bin/bash
```

```
read nom prenom
echo "Bonjour $nom $prenom !"
```

Code : Console

```
Deschamps Mathieu
Bonjour Deschamps Mathieu !
```



`read` lit ce que vous tapez mot par mot (en considérant que les mots sont séparés par des espaces). Il assigne chaque mot à une variable différente, d'où le fait que le nom et le prénom ont été correctement et respectivement assignés à `$nom` et `$prenom`.
Si vous rentrez plus de mots au clavier que vous n'avez prévu de variables pour en stocker, la dernière variable de la liste récupèrera tous les mots restants. En clair, si j'avais tapé pour le programme précédent « Nebra Mathieu Cyril », la variable `$prenom` aurait eu pour valeur « Mathieu Cyril ».

-p : afficher un message de prompt

Bon : notre programme n'est pas très clair et nous devrions afficher un message pour que l'utilisateur sache quoi faire. Avec l'option `-p` de `read`, vous pouvez faire cela :

Code : Console

```
#!/bin/bash

read -p 'Entrez votre nom : ' nom
echo "Bonjour $nom !"
```



Notez que le message `'Entrez votre nom'` a été entouré de quotes. Si on ne l'avait pas fait, le bash aurait considéré que chaque mot était un paramètre différent !

Résultat :

Code : Console

```
Entrez votre nom : Mathieu
Bonjour Mathieu !
```

C'est mieux !

-n : limiter le nombre de caractères

Avec `-n`, vous pouvez au besoin couper au bout de X caractères si vous ne voulez pas que l'utilisateur insère un message trop long.

Exemple :

Code : Console

```
#!/bin/bash

read -p 'Entrez votre login (5 caractères max) : ' -n 5 nom
echo "Bonjour $nom !"
```

Code : Console

```
Entrez votre login (5 caractères max) : mathiBonjour mathi !
```

Notez que le bash coupe automatiquement au bout de 5 caractères sans que vous ayez besoin d'appuyer sur la touche `Entrée`. Ce n'est pas très esthétique du coup, parce que le message s'affiche sur la même ligne. Pour éviter cela, vous pouvez faire un `echo` avec des `\n`, comme vous avez appris à le faire plus tôt :

Code : Console

```
#!/bin/bash

read -p 'Entrez votre login (5 caractères max) : ' -n 5 nom
echo -e "\nBonjour $nom !"
```

Code : Console

```
Entrez votre login (5 caractères max) : mathi
Bonjour mathi !
```

`-t` : limiter le temps autorisé pour saisir un message

Vous pouvez définir un *timeout* avec `-t`, c'est-à-dire un nombre de secondes au bout duquel le `read` s'arrêtera.

Code : Console

```
#!/bin/bash

read -
p 'Entrez le code de désamorçage de la bombe (vous avez 5 secondes) : '  -
t 5 code
echo -e "\nBoum !"
```

`-s` : ne pas afficher le texte saisi

Probablement plus utile, le paramètre `-s` masque les caractères que vous saisissez. Cela vous servira notamment si vous souhaitez que l'utilisateur entre un mot de passe :

Code : Console

```
#!/bin/bash

read -p 'Entrez votre mot de passe : ' -s pass
echo -
```

```
e "\nMerci ! Je vais dire à tout le monde que votre mot de passe est $pass ! :)"
```

Code : Console

```
Entrez votre mot de passe :
Merci ! Je vais dire à tout le monde que votre mot de passe est supertopsecret38 !
```

Comme vous pouvez le constater, le mot de passe que j'ai entré ne s'affiche pas lors de l'instruction `read`.

Effectuer des opérations mathématiques

En bash, les variables sont toutes des chaînes de caractères. En soi, le bash n'est pas vraiment capable de manipuler des nombres ; il n'est donc pas capable d'effectuer des opérations.

Heureusement, il est possible de passer par des commandes (eh oui, encore). Ici, la commande à connaître est `let`.

Code : Console

```
let "a = 5"
let "b = 2"
let "c = a + b"
```

À la fin de ce script, la variable `$c` vaudra 7. Testons :

Code : Console

```
#!/bin/bash

let "a = 5"
let "b = 2"
let "c = a + b"
echo $c
```

Code : Console

```
7
```

Les opérations utilisables sont :

- l'addition : + ;
- la soustraction : - ;
- la multiplication : * ;
- la division : / ;
- la puissance : ** ;
- le modulo (renvoie le reste de la division entière) : %.

Quelques exemples :

Code : Console

```
let "a = 5 * 3" # $a = 15
let "a = 4 ** 2" # $a = 16 (4 au carré)
let "a = 8 / 2" # $a = 4
```

```
let "a = 10 / 3" # $a = 3
let "a = 10 % 3" # $a = 1
```

Une petite explication pour les deux dernières lignes :

- 10 / 3 = 3 car la division est entière (la commande ne renvoie pas de nombres décimaux) ;
- 10 % 3 renvoie 1 car le reste de la division de 10 par 3 est 1. En effet, 3 « rentre » 3 fois dans 10 (ça fait 9), et il reste 1 pour aller à 10.

Notez qu'il est possible aussi de contracter les commandes, comme cela se fait en langage C.
Ainsi :

Code : Console

```
let "a = a * 3"
```

… équivaut à écrire :

Code : Console

```
let "a *= 3"
```



Actuellement, les résultats renvoyés sont des nombres entiers et non des nombres décimaux. Si vous voulez travailler avec des nombres décimaux, renseignez-vous sur le fonctionnement de la commande `bc`.

Les variables d'environnement

Actuellement, les variables que vous créez dans vos scripts bash n'existent que dans ces scripts. En clair, une variable définie dans un programme A ne sera pas utilisable dans un programme B.

Les variables d'environnement sont des variables que l'on peut utiliser dans n'importe quel programme. On parle aussi parfois de **variables globales**. Vous pouvez afficher toutes celles que vous avez actuellement en mémoire avec la commande `env` :

Code : Console

```
$ env
ORBIT_SOCKETDIR=/tmp/orbit-mateo21
GLADE_PIXMAP_PATH=:/usr/share/glade3/pixmaps
TERM=xterm
SHELL=/bin/bash
GTK_MODULES=canberra-gtk-module
USER=mateo21
PATH=/home/mateo21/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin: /usr/bin:/sbin:/bi
GDM_XSERVER_LOCATION=local
PWD=/home/mateo21/bin
EDITOR=nano
SHLVL=1
HOME=/home/mateo21
OLDPWD=/home/mateo21

[ ... ]
```

Il y en a beaucoup. Certaines sont très utiles, d'autres moins. Parmi celles que je peux vous commenter et qui peuvent s'avérer utiles, on trouve :

- `SHELL` : indique quel type de shell est en cours d'utilisation (sh, bash, ksh…) ;

- PATH : une liste des répertoires qui contiennent des exécutables que vous souhaitez pouvoir lancer sans indiquer leur répertoire. Nous en avons parlé un peu plus tôt. Si un programme se trouve dans un de ces dossiers, vous pourrez l'invoquer quel que soit le dossier dans lequel vous vous trouvez ;
- EDITOR : l'éditeur de texte par défaut qui s'ouvre lorsque cela est nécessaire ;
- HOME : la position de votre dossier home ;
- PWD : le dossier dans lequel vous vous trouvez ;
- OLDPWD : le dossier dans lequel vous vous trouviez auparavant.



Notez que les noms de ces variables sont, par convention, écrits en majuscules.

Comment utiliser ces variables dans vos scripts ? C'est très simple, il suffit de les appeler par leur nom !

Exemple :

Code : Console

```
#!/bin/bash

echo "Votre éditeur par défaut est $EDITOR"
```

Code : Console

```
Votre éditeur par défaut est nano
```



Plus rarement, vous pourriez avoir besoin de définir votre propre variable d'environnement. Pour cela, on utilise la commande export que vous avez pu voir dans votre .bashrc.

Les variables des paramètres

Comme toutes les commandes, vos scripts bash peuvent eux aussi accepter des paramètres. Ainsi, on pourrait appeler notre script comme ceci :

Code : Console

```
./variables.sh param1 param2 param3
```

Le problème, c'est que nous n'avons toujours pas vu comment récupérer ces paramètres dans notre script. Pourtant, c'est très simple à réaliser !

En effet, des variables sont automatiquement créées :

- $# : contient le nombre de paramètres ;
- $0 : contient le nom du script exécuté (ici ./variables.sh) ;
- $1 : contient le premier paramètre ;
- $2 : contient le second paramètre ;
- … ;
- **$9** : contient le 9e paramètre.

Essayons :

Code : Console

```
#!/bin/bash

echo "Vous avez lancé $0, il y a $# paramètres"
```

```
echo "Le paramètre 1 est $1"
```

Code : Console

```
$ ./variables.sh param1 param2 param3
Vous avez lancé ./variables.sh, il y a 3 paramètres
Le paramètre 1 est param1
```



Et si on utilise plus de neuf paramètres ? J'ai cru voir que les variables s'arrêtaient à `$9`…

Là, ça va un peu loin, mais ça peut arriver. On peut imaginer un script qui accepte une liste de fichiers en paramètre. Rien ne nous empêcherait de lui envoyer quinze paramètres dans ce cas :

Code : Console

```
./script.sh fichier1 fichier2 fichier3 fichier4 ... fichier14 fichier15
```

En général, pour traiter autant de paramètres, on s'occupera d'eux un par un… On peut « décaler » les paramètres dans les variables `$1`, `$2`, etc. à l'aide de la commande `shift`.

Reprenons notre script :

Code : Console

```
#!/bin/bash

echo "Le paramètre 1 est $1"
shift
echo "Le paramètre 1 est maintenant $1"
```

Code : Console

```
$ ./variables.sh param1 param2 param3
Le paramètre 1 est param1
Le paramètre 1 est maintenant param2
```

Comme vous le voyez, les paramètres ont été décalés : `$1` correspond après le `shift` au second paramètre, `$2` au troisième paramètre, etc.

Bien sûr, `shift` est généralement utilisé dans une boucle qui permet de traiter les paramètres un par un. Nous verrons d'ailleurs comment faire des boucles dans peu de temps.

Les tableaux

Le bash gère également les variables « tableaux ». Ce sont des variables qui contiennent plusieurs cases, comme un tableau. Vous en aurez probablement besoin un jour ; voyons comment cela fonctionne.

Pour définir un tableau, on peut faire comme ceci :

Code : Console

```
tableau=('valeur0' 'valeur1' 'valeur2')
```

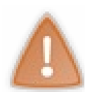
Cela crée une variable `tableau` qui contient trois valeurs (`valeur0`, `valeur1`, `valeur2`).

Pour accéder à une case du tableau, il faut utiliser la syntaxe suivante :

Code : Console

```
${tableau[2]}
```

… ceci affichera le contenu de la case n° 2 (donc `valeur2`).



Les cases sont numérotées à partir de 0 ! La première case a donc le numéro 0.
Notez par ailleurs que pour afficher le contenu d'une case du tableau, vous devez entourer votre variable d'accolades comme je l'ai fait pour `${tableau[2]}`.

Vous pouvez aussi définir manuellement le contenu d'une case :

Code : Console

```
tableau[2]='valeur2'
```

Essayons tout ceci dans un script :

Code : Console

```
#!/bin/bash

tableau=('valeur0' 'valeur1' 'valeur2')
tableau[5]='valeur5'
echo ${tableau[1]}
```

À votre avis, que va afficher ce script ?

Réponse :

Code : Console

```
valeur1
```



Comme vous pouvez le constater, le tableau peut avoir autant de cases que vous le désirez. La numérotation n'a pas besoin d'être continue, vous pouvez sauter des cases sans aucun problème (la preuve, il n'y a pas de case n° 3 ni de case n° 4 dans mon script précédent).

Vous pouvez afficher l'ensemble du contenu du tableau d'un seul coup en utilisant `${tableau[*]}` :

Code : Console

```
#!/bin/bash

tableau=('valeur0' 'valeur1' 'valeur2')
tableau[5]='valeur5'
echo ${tableau[*]}
```

Code : Console

```
valeur0 valeur1 valeur2 valeur5
```

En résumé

- Comme dans la plupart des langages de programmation, on peut créer des variables en shell qui stockent temporairement des valeurs en mémoire. Une variable nommée `variable` est accessible en écrivant `$variable`.
- La commande `echo` affiche un texte ou le contenu d'une variable dans la console.
- `read` attend une saisie au clavier de la part de l'utilisateur et stocke le résultat dans une variable.
- On peut effectuer des opérations mathématiques sur des nombres à l'aide de la commande `let`.
- Certaines variables sont accessibles partout, dans tous les scripts : ce sont les variables d'environnement. On peut les lister avec la commande `env`.
- Les paramètres envoyés à notre script (comme `./script -p`) sont transmis dans des variables numérotées : `$1`, `$2`, `$3`… Le nombre de paramètres envoyés est indiqué dans la variable `$#`.

 Les conditions

La prise de décision est un élément indispensable dans tout programme. Si on ne pouvait pas décider quoi faire, le programme ferait toujours la même chose… ce qui serait bien ennuyeux.

Les branchements conditionnels (que nous abrègerons « conditions ») constituent un moyen de dire dans notre script « SI cette variable vaut tant, ALORS fais ceci, SINON fais cela ».
Si vous connaissez déjà un autre langage de programmation, cela doit vous être familier. Sinon, ne vous en faites pas, vous allez très vite comprendre le concept.

if : la condition la plus simple

Le type de condition le plus courant est le `if`, qui signifie « si ».

Si

Les conditions ont la forme suivante :

Code : Console

```
SI test_de_variable
ALORS
------> effectuer_une_action
FIN SI
```

Bien entendu, ce n'est pas du bash. Il s'agit juste d'un schéma pour vous montrer quelle est la forme d'une condition.

La syntaxe en bash est la suivante :

Code : Console

```
if [ test ]
then
        echo "C'est vrai"
fi
```

Le mot `fi` (`if` à l'envers !) à la fin indique que le `if` s'arrête là. Tout ce qui est entre le `then` et le `fi` sera exécuté uniquement si le test est vérifié.



Vous noterez — c'est très important — qu'il y a des espaces à l'intérieur des crochets. On ne doit pas écrire `[test]` mais `[ test ]` !



Il existe une autre façon d'écrire le `if` : en plaçant le `then` sur la même ligne. Dans ce cas, il ne faut pas oublier de rajouter un point-virgule après les crochets :

Code : Console

```
if [ test ]; then
        echo "C'est vrai"
fi
```

À la place du mot `test`, il faut indiquer votre test. C'est à cet endroit que vous testerez la valeur d'une variable, par exemple. Ici, nous allons voir un cas simple où nous testons la valeur d'une chaîne de caractères, puis nous apprendrons à faire des tests plus compliqués un peu plus loin dans le chapitre.

Faisons quelques tests sur un script que nous appellerons `conditions.sh` :

Code : Console

```
#!/bin/bash

nom="Bruno"

if [ $nom = "Bruno" ]
then
        echo "Salut Bruno !"
fi
```

Comme `$nom` est bien égal à « Bruno », ce script affichera :

Code : Console

```
Salut Bruno !
```

Essayez de changer le test : si vous n'écrivez pas précisément « Bruno », le `if` ne sera pas exécuté et votre script n'affichera donc rien.

Notez aussi que vous pouvez tester deux variables à la fois dans le `if` :

Code : Console

```
#!/bin/bash

nom1="Bruno"
nom2="Marcel"

if [ $nom1 = $nom2 ]
then
        echo "Salut les jumeaux !"
fi
```

Comme ici `$nom1` est différent de `$nom2`, le contenu du `if` ne sera pas exécuté. Le script n'affichera donc rien.

Sinon

Si vous souhaitez faire quelque chose de particulier quand la condition n'est **pas** remplie, vous pouvez rajouter un `else` qui signifie « sinon ».

En français, cela s'écrirait comme ceci :

Code : Console

```
SI test_de_variable
ALORS
------> effectuer_une_action
SINON
------> effectuer_une_action
FIN SI
```

Code : Console

```
if [ test ]
then
        echo "C'est vrai"
else
        echo "C'est faux"
fi
```

Reprenons notre script de tout à l'heure et ajoutons-lui un `else` :

Code : Console

```
#!/bin/bash

nom="Bruno"

if [ $nom = "Bruno" ]
then
        echo "Salut Bruno !"
else
        echo "J'te connais pas, ouste !"
fi
```

Bon : comme la variable vaut toujours la même chose, le `else` ne sera jamais exécuté, ce n'est pas rigolo. Je vous propose plutôt de vous baser sur le premier paramètre (`$1`) envoyé au script :

Code : Console

```
#!/bin/bash

if [ $1 = "Bruno" ]
then
        echo "Salut Bruno !"
else
        echo "J'te connais pas, ouste !"
fi
```

Testez maintenant votre script en lui donnant un paramètre :

Code : Console

```
$ ./conditions.sh Bruno
Salut Bruno !
```

Et si vous mettez autre chose :

Code : Console

```
$ ./conditions.sh Jean
J'te connais pas, ouste !
```



Notez que le script plante si vous oubliez de l'appeler avec un paramètre. Pour bien faire, il faudrait d'abord vérifier dans un `if` s'il y a au moins un paramètre. Nous apprendrons à faire cela plus loin.

Sinon si

Il existe aussi le mot clé `elif`, abréviation de « else if », qui signifie « sinon si ». Sa forme ressemble à ceci :

Code : Console

```
SI test_de_variable
ALORS
------> effectuer_une_action
SINON SI autre_test
ALORS
------> effectuer_une_action
SINON SI encore_un_autre_test
ALORS
------> effectuer_une_action
SINON
------> effectuer_une_action
FIN SI
```

C'est un peu plus compliqué, n'est-ce pas ?

Sachez que l'on peut mettre autant de « sinon si » que l'on veut ; là, j'en ai mis deux. En revanche, on ne peut mettre qu'un seul « sinon », qui sera exécuté à la fin si aucune des conditions précédentes n'est vérifiée.

Bash va d'abord analyser le premier test. S'il est vérifié, il effectuera la première action indiquée ; s'il ne l'est pas, il ira au premier « sinon si », au second, etc., jusqu'à trouver une condition qui soit vérifiée. Si aucune condition ne l'est, c'est le « sinon » qui sera lu.

Bien ! Voyons comment cela s'écrit en bash :

Code : Console

```
if [ test ]
then
        echo "Le premier test a été vérifié"
elif [ autre_test ]
then
        echo "Le second test a été vérifié"
elif [ encore_autre_test ]
then
        echo "Le troisième test a été vérifié"
else
        echo "Aucun des tests précédents n'a été vérifié"
fi
```

On peut reprendre notre script précédent et l'adapter pour utiliser des `elif` :

Code : Console

```
#!/bin/bash

if [ $1 = "Bruno" ]
then
        echo "Salut Bruno !"
elif [ $1 = "Michel" ]
then
        echo "Bien le bonjour Michel"
elif [ $1 = "Jean" ]
```

```
then
        echo "Hé Jean, ça va ?"
else
        echo "J'te connais pas, ouste !"
fi
```

Vous pouvez tester ce script ; encore une fois, n'oubliez pas d'envoyer un paramètre sinon il plantera, ce qui est normal.

Les tests

Voyons maintenant un peu quels sont les tests que nous pouvons faire. Pour l'instant, on a juste vérifié si deux chaînes de caractères étaient identiques, mais on peut faire beaucoup plus de choses que cela !

Les différents types de tests

Il est possible d'effectuer trois types de tests différents en bash :

- des tests sur des chaînes de caractères ;
- des tests sur des nombres ;
- des tests sur des fichiers.

Nous allons maintenant découvrir tous ces types de tests et les essayer. :)

Tests sur des chaînes de caractères

Comme vous devez désormais le savoir, en bash toutes les variables sont considérées comme des chaînes de caractères. Il est donc très facile de tester ce que vaut une chaîne de caractères. Vous trouverez les différents types de tests disponibles sur le tableau suivante.

Vérifions par exemple si deux paramètres sont différents :

Code : Console

```
#!/bin/bash

if [ $1 != $2 ]
then
        echo "Les 2 paramètres sont différents !"
else
        echo "Les 2 paramètres sont identiques !"
fi
```

Code : Console

```
$ ./conditions.sh Bruno Bernard
Les 2 paramètres sont différents !
```

Code : Console

```
$ ./conditions.sh Bruno Bruno
Les 2 paramètres sont identiques !
```

Les différents types de tests sur des chaînes

Condition	Signification
`$chaine1 = $chaine2`	Vérifie si les deux chaînes sont identiques. Notez que bash est sensible à la casse : « b » est donc différent de « B ». Il est aussi possible d'écrire « == » pour les habitués du langage C.
`$chaine1 != $chaine2`	Vérifie si les deux chaînes sont différentes.
`-z $chaine`	Vérifie si la chaîne est vide.
`-n $chaine`	Vérifie si la chaîne est non vide.

On peut aussi vérifier si le paramètre existe avec `-z` (vérifie si la chaîne est vide). En effet, si une variable n'est pas définie, elle est considérée comme vide par bash. On peut donc par exemple s'assurer que `$1` existe en faisant comme suit :

Code : Console

```
#!/bin/bash

if [ -z $1 ]
then
        echo "Pas de paramètre"
else
        echo "Paramètre présent"
fi
```

Code : Console

```
$ ./conditions.sh
Pas de paramètre
```

Code : Console

```
$ ./conditions.sh param
Paramètre présent
```

Tests sur des nombres

Bien que bash gère les variables comme des chaînes de caractères pour son fonctionnement interne, rien ne nous empêche de faire des comparaisons de nombres si ces variables en contiennent. Vous trouverez les différents types de tests disponibles sur le tableau suivante.

Les différents types de tests sur des nombres

Condition	Signification
`$num1 -eq $num2`	Vérifie si les nombres sont égaux (**eq***ual*). À ne pas confondre avec le « = » qui, lui, compare deux chaînes de caractères.
`$num1 -ne $num2`	Vérifie si les nombres sont différents (**n***on* **e***qual*). Encore une fois, ne confondez pas avec « != » qui est censé être utilisé sur des chaînes de caractères.
`$num1 -lt $num2`	Vérifie si `num1` est inférieur (<) à `num2` (**l***ower* **t***han*).
`$num1 -le $num2`	Vérifie si `num1` est inférieur ou égal (<=) à `num2` (**l***ower or* **e***qual*).

`$num1 -gt $num2`	Vérifie si `num1` est supérieur (>) à `num2` (**g**reater **t**han).
`$num1 -ge $num2`	Vérifie si `num1` est supérieur ou égal (>=) à `num2` (**g**reater or **e**qual).

Vérifions par exemple si un nombre est supérieur ou égal à un autre nombre :

Code : Console

```
#!/bin/bash

if [ $1 -ge 20 ]
then
        echo "Vous avez envoyé 20 ou plus"
else
        echo "Vous avez envoyé moins de 20"
fi
```

Code : Console

```
$ ./conditions.sh 23
Vous avez envoyé 20 ou plus
```

Code : Console

```
$ ./conditions.sh 11
Vous avez envoyé moins de 20
```

Tests sur des fichiers

Un des avantages de bash sur d'autres langages est que l'on peut très facilement faire des tests sur des fichiers : savoir s'ils existent, si on peut écrire dedans, s'ils sont plus vieux, plus récents, etc. Le tableau suivante présente les différents types de tests disponibles.

Les différents types de tests sur des fichiers

Condition	Signification
`-e $nomfichier`	Vérifie si le fichier existe.
`-d $nomfichier`	Vérifie si le fichier est un répertoire. N'oubliez pas que sous Linux, tout est considéré comme un fichier, même un répertoire !
`-f $nomfichier`	Vérifie si le fichier est un… fichier. Un vrai fichier cette fois, pas un dossier.
`-L $nomfichier`	Vérifie si le fichier est un lien symbolique (raccourci).
`-r $nomfichier`	Vérifie si le fichier est lisible (r).
`-w $nomfichier`	Vérifie si le fichier est modifiable (w).
`-x $nomfichier`	Vérifie si le fichier est exécutable (x).
`$fichier1 -nt $fichier2`	Vérifie si `fichier1` est plus récent que `fichier2` (**n**ewer **t**han).
`$fichier1`	

$fichier1 -ot $fichier2	Vérifie si `fichier1` est plus vieux que `fichier2` (***o**lder **t**han*).

Je vous propose de faire un script qui demande à l'utilisateur d'entrer le nom d'un répertoire et qui vérifie si c'en est bien un :

Code : Console

```
#!/bin/bash

read -p 'Entrez un répertoire : ' repertoire

if [ -d $repertoire ]
then
        echo "Bien, vous avez compris ce que j'ai dit !"
else
        echo "Vous n'avez rien compris..."
fi
```

Code : Console

```
Entrez un répertoire : /home
Bien, vous avez compris ce que j'ai dit !
```

Code : Console

```
Entrez un répertoire : rienavoir.txt
Vous n'avez rien compris...
```

Notez que bash vérifie au préalable que le répertoire existe bel et bien.

Effectuer plusieurs tests à la fois

Dans un `if`, il est possible de faire plusieurs tests à la fois. En général, on vérifie :

- si un test est vrai **ET** qu'un autre test est vrai ;
- si un test est vrai **OU** qu'un autre test est vrai.

Les deux symboles à connaître sont :

- **&&** : signifie « et » ;
- || : signifie « ou ».

Il faut encadrer chaque condition par des crochets. Prenons un exemple :

Code : Console

```
#!/bin/bash

if [ $# -ge 1 ] && [ $1 = 'koala' ]
then
        echo "Bravo !"
        echo "Vous connaissez le mot de passe"
else
        echo "Vous n'avez pas le bon mot de passe"
fi
```

Le test vérifie deux choses :

- qu'il y a au moins un paramètre (« si `$#` est supérieur ou égal à 1 ») ;
- que le premier paramètre est bien `koala` (« si `$1` est égal à `koala` »).

Si ces deux conditions sont remplies, alors le message indiquant que l'on a trouvé le bon mot de passe s'affichera.

Code : Console

```
$ ./conditions.sh koala
Bravo !
Vous connaissez le mot de passe
```



Notez que les tests sont effectués l'un après l'autre et seulement s'ils sont nécessaires. Bash vérifie d'abord s'il y a au moins un paramètre. Si ce n'est pas le cas, il ne fera pas le second test puisque la condition ne sera de toute façon pas vérifiée.

Inverser un test

Il est possible d'inverser un test en utilisant la négation. En bash, celle-ci est exprimée par le point d'exclamation « `!` ».

Code : Console

```
if [ ! -e fichier ]
then
        echo "Le fichier n'existe pas"
fi
```

Vous en aurez besoin, donc n'oubliez pas ce petit point d'exclamation.

case : tester plusieurs conditions à la fois

On a vu tout à l'heure un `if` un peu complexe qui faisait appel à des `elif` et à un `else` :

Code : Console

```
#!/bin/bash

if [ $1 = "Bruno" ]
then
        echo "Salut Bruno !"
elif [ $1 = "Michel" ]
then
        echo "Bien le bonjour Michel"
elif [ $1 = "Jean" ]
then
        echo "Hé Jean, ça va ?"
else
        echo "J'te connais pas, ouste !"
fi
```

Ce genre de « gros `if` qui teste toujours la même variable » ne pose pas de problème mais n'est pas forcément très facile à lire pour le programmeur. À la place, il est possible d'utiliser l'instruction `case` si nous voulons.

Le rôle de `case` est de tester la valeur d'une même variable, mais de manière plus concise et lisible.

Voyons comment on écrirait la condition précédente avec un `case` :

Code : Console

```
#!/bin/bash

case $1 in
        "Bruno")
                echo "Salut Bruno !"
                ;;
        "Michel")
                echo "Bien le bonjour Michel"
                ;;
        "Jean")
                echo "Hé Jean, ça va ?"
                ;;
        *)
                echo "J'te connais pas, ouste !"
                ;;
esac
```

Cela fait beaucoup de nouveautés d'un coup.
Analysons la structure du `case` !

Code : Console

```
case $1 in
```

Tout d'abord, on indique que l'on veut tester la valeur de la variable `$1`. Bien entendu, vous pouvez remplacer `$1` par n'importe quelle variable que vous désirez tester.

Code : Console

```
"Bruno")
```

Là, on teste une valeur. Cela signifie « Si `$1` est égal à Bruno ». Notez que l'on peut aussi utiliser une étoile comme joker : « `B*` » acceptera tous les mots qui commencent par un B majuscule.

Si la condition est vérifiée, tout ce qui suit est exécuté jusqu'au prochain double point-virgule :

Code : Console

```
;;
```

Important, il ne faut pas l'oublier : le double point-virgule dit à bash d'arrêter là la lecture du `case`. Il saute donc à la ligne qui suit le `esac` signalant la fin du `case`.

Code : Console

```
*)
```

C'est en fait le « else » du `case`. Si aucun des tests précédents n'a été vérifié, c'est alors cette section qui sera lue.

Code : Console

```
esac
```

Marque la fin du `case` (`esac`, c'est « case » à l'envers !).

Nous pouvons aussi faire des « ou » dans un `case`. Dans ce cas, petit piège, il ne faut pas mettre deux `||` mais un seul !
Exemple :

Code : Console

```
#!/bin/bash

case $1 in
        "Chien" | "Chat" | "Souris")
                echo "C'est un mammifère"
                ;;
        "Moineau" | "Pigeon")
                echo "C'est un oiseau"
                ;;
        *)
                echo "Je ne sais pas ce que c'est"
                ;;
esac
```

En résumé

- On effectue des tests dans ses programmes grâce aux `if`, `elif`, `else`, `fi`.
- On peut comparer deux chaînes de caractères entre elles, mais aussi des nombres. On peut également effectuer des tests sur des fichiers : est-ce que celui-ci existe ? Est-il exécutable ? Etc.
- Au besoin, il est possible de combiner plusieurs tests à la fois avec les symboles `&&` (ET) et `||` (OU).
- Le symbole `!` (point d'exclamation) exprime la négation dans une condition.
- Lorsque l'on effectue beaucoup de tests sur une même variable, il est parfois plus pratique d'utiliser un bloc `case in… esac` plutôt qu'un bloc `if… fi`.



Les boucles

Nousallons découvrir dans ce chapitre un autre élément de base de tous les langages : les boucles. Ces structures permettent de répéter autant de fois que nécessaire une partie du code. En bash, on n'y échappe pas !

Les consignes sont les mêmes que pour le chapitre sur les conditions : il faut être vigilant sur la syntaxe. Une espace de trop ou de moins, l'oubli d'un caractère spécial et plus rien ne fonctionne. Soyez donc très rigoureux lorsque vous codez !

Si vous suivez cette simple règle, vous n'aurez pas de problèmes.

while : boucler « tant que »

Le type de boucle que l'on rencontre le plus couramment en bash est `while`.

Le principe est de faire un code qui ressemble à ceci :

Code : Console

```
TANT QUE test
FAIRE
------> effectuer_une_action
RECOMMENCER
```

En bash, on l'écrit comme ceci :

Code : Console

```
while [ test ]
do
        echo 'Action en boucle'
done
```



Il est aussi possible, comme pour le `if`, d'assembler les deux premières lignes en une, à condition de mettre un point-virgule :

Code : Console

```
while [ test ]; do
echo 'Action en boucle'
done
```

On va demander à l'utilisateur de dire « oui » et répéter cette action tant qu'il n'a pas fait ce que l'on voulait. Nous allons créer un script `boucles.sh` pour l'occasion :

Code : Console

```
#!/bin/bash

while [ -z $reponse ] || [ $reponse != 'oui' ]
do
        read -p 'Dites oui : ' reponse
done
```

On fait deux tests.

1. Est-ce que `$reponse` est vide ?

2. Est-ce que `$reponse` est différent de `oui` ?

Comme il s'agit d'un OU (`||`), tant que l'un des deux tests est vrai, on recommence la boucle. Cette dernière pourrait se traduire par : « Tant que la réponse est vide ou que la réponse est différente de `oui` ».
Nous sommes obligés de vérifier d'abord si la variable n'est pas vide, car si elle l'est, le second test plante (essayez, vous verrez).

Essayons ce script :

Code : Console

```
Dites oui : euh
Dites oui : non
Dites oui : bon
Dites oui : oui
```

Comme vous pouvez le voir, il ne s'arrête que lorsque l'on a tapé `oui` !



Il existe aussi le mot clé `until`, qui est l'exact inverse de `while`. Il signifie « Jusqu'à ce que ».
Remplacez juste `while` par `until` dans le code précédent pour l'essayer.

for : boucler sur une liste de valeurs



Avertissement pour ceux qui ont déjà fait de la programmation : le `for` en bash ne se comporte pas de la même manière que le `for` auquel vous êtes habitués dans un autre langage, comme le C ou le PHP. Lisez donc attentivement.

Parcourir une liste de valeurs

La boucle `for` permet de parcourir une liste de valeurs et de boucler autant de fois qu'il y a de valeurs.

Concrètement, la forme d'un `for` est la suivante :

Code : Console

```
POUR variable PRENANT valeur1 valeur2 valeur3
FAIRE
------> effectuer_une_action
VALEUR_SUIVANTE
```

La variable va prendre successivement les valeurs `valeur1`, `valeur2`, `valeur3`. La boucle va donc être exécutée trois fois et la variable vaudra à chaque fois une nouvelle valeur de la liste.

En bash, la boucle `for` s'écrit comme ceci :

Code : Console

```
#!/bin/bash

for variable in 'valeur1' 'valeur2' 'valeur3'
do
        echo "La variable vaut $variable"
done
```

Ce qui donne, si on l'exécute :

Code : Console

```
La variable vaut valeur1
La variable vaut valeur2
La variable vaut valeur3
```

Vous pouvez donc vous servir du `for` pour faire une boucle sur une liste de valeurs que vous définissez :

Code : Console

```
#!/bin/bash

for animal in 'chien' 'souris' 'moineau'
do
        echo "Animal en cours d'analyse : $animal"
done
```

Code : Console

```
Animal en cours d'analyse : chien
Animal en cours d'analyse : souris
Animal en cours d'analyse : moineau
```

Toutefois, la liste de valeurs n'a pas besoin d'être définie directement dans le code. On peut utiliser une variable :

Code : Console

```
#!/bin/bash

liste_fichiers=`ls`

for fichier in $liste_fichiers
do
        echo "Fichier trouvé : $fichier"
done
```

Ce script liste tous les fichiers trouvés dans le répertoire actuel :

Code : Console

```
Fichier trouvé : boucles.sh
Fichier trouvé : conditions.sh
Fichier trouvé : variables.sh
```

On pourrait faire un code plus court sans passer par une variable `$liste_fichiers` en écrivant :

Code : Console

```
#!/bin/bash
for fichier in `ls`
do
        echo "Fichier trouvé : $fichier"
done
```

Bien entendu, ici, on ne fait qu'afficher le nom du fichier, ce qui n'est ni très amusant ni très utile. On pourrait se servir de notre script pour renommer chacun des fichiers du répertoire actuel en leur ajoutant un suffixe `-old` par exemple :

Code : Console

```
#!/bin/bash

for fichier in `ls`
do
        mv $fichier $fichier-old
done
```

Essayons de voir si l'exécution du script renomme bien tous les fichiers :

Code : Console

```
$ ls
boucles.sh  conditions.sh  variables.sh
$ ./boucles.sh
$ ls
boucles.sh-old  conditions.sh-old  variables.sh-old
```

À vous de jouer ! Essayez de créer un script `multirenommage.sh`, reposant sur ce principe, qui va rajouter le suffixe `-old`… uniquement aux fichiers qui correspondent au paramètre envoyé par l'utilisateur !

Code : Console

```
./multirenommage.sh *.txt
```

Si aucun paramètre n'est envoyé, vous demanderez à l'utilisateur de saisir le nom des fichiers à renommer avec `read`.

Un `for` plus classique

Pour les habitués d'autres langages de programmation, le `for` est une boucle qui permet de faire prendre à une variable une suite de nombres.

En bash, comme on l'a vu, le `for` permet de parcourir une liste de valeurs. Toutefois, en trichant un peu à l'aide de la commande `seq`, il est possible de simuler un `for` classique :

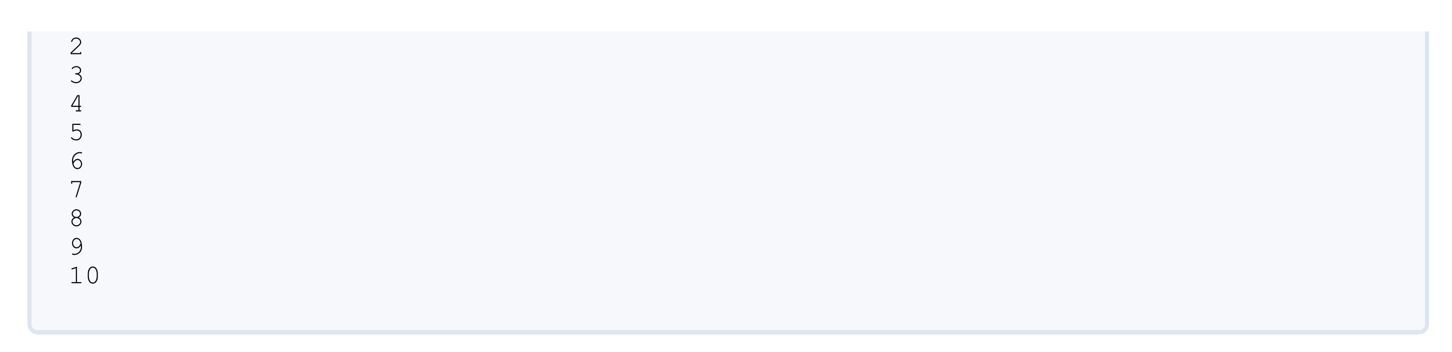
Code : Console

```
#!/bin/bash
for i in `seq 1 10`;
do
        echo $i
done
```

Explication : `seq` génère tous les nombres allant du premier paramètre au dernier paramètre, donc 1 2 3 4 5 6 7 8 9 10.

Code : Console

```
1
```



```
2
3
4
5
6
7
8
9
10
```

Si vous le voulez, vous pouvez changer le pas et avancer de deux en deux par exemple. Dans ce cas, il faut écrire `seq 1 2 10` pour aller de 1 à 10 en avançant de deux en deux ; cela va donc générer les nombres 1 3 5 7 9.

En résumé

- Pour exécuter une série de commandes plusieurs fois, on utilise des boucles.
- `while` permet de boucler tant qu'une condition est remplie. Le fonctionnement des conditions dans les boucles est le même que celui des blocs `if` découverts dans le chapitre précédent.
- `for` permet de boucler sur une série de valeurs définies. À l'intérieur de la boucle, une variable prend successivement les valeurs indiquées.

 TP : générateur de galerie d'images

L'intérêtdu bash ne commence à se faire sentir que lorsque l'on code de vrais scripts, alors… il est grand temps de pratiquer !

Dans ce TP, vous allez devoir réutiliser un peu tout ce que vous avez appris jusqu'ici sur bash et sur Linux en général. N'oubliez pas que dans les scripts bash vous pouvez réutiliser toutes les commandes de la console que vous connaissez : `ls`, `grep`, `cut`, `sort`, les flux… allez-y, tous les coups sont permis. Vous risquez même d'avoir à lire le manuel pour trouver quelques paramètres !

Votre objectif est de créer une page web présentant une galerie d'images en fonction des fichiers présents dans un dossier. Plus facile à dire qu'à faire, car vous allez voir qu'il y a là un vrai défi. Bonne chance à tous.

Objectifs

Nous souhaitons réaliser dans ce TP un **générateur de galerie d'images** en bash.

Le script s'appellera `galerie.sh`. Pour sa première version, il faudra le placer dans un dossier contenant des images ; il générera des miniatures à partir de ces dernières et un fichier HTML présentant toutes les images du dossier.

Concrètement, le script devra donc :

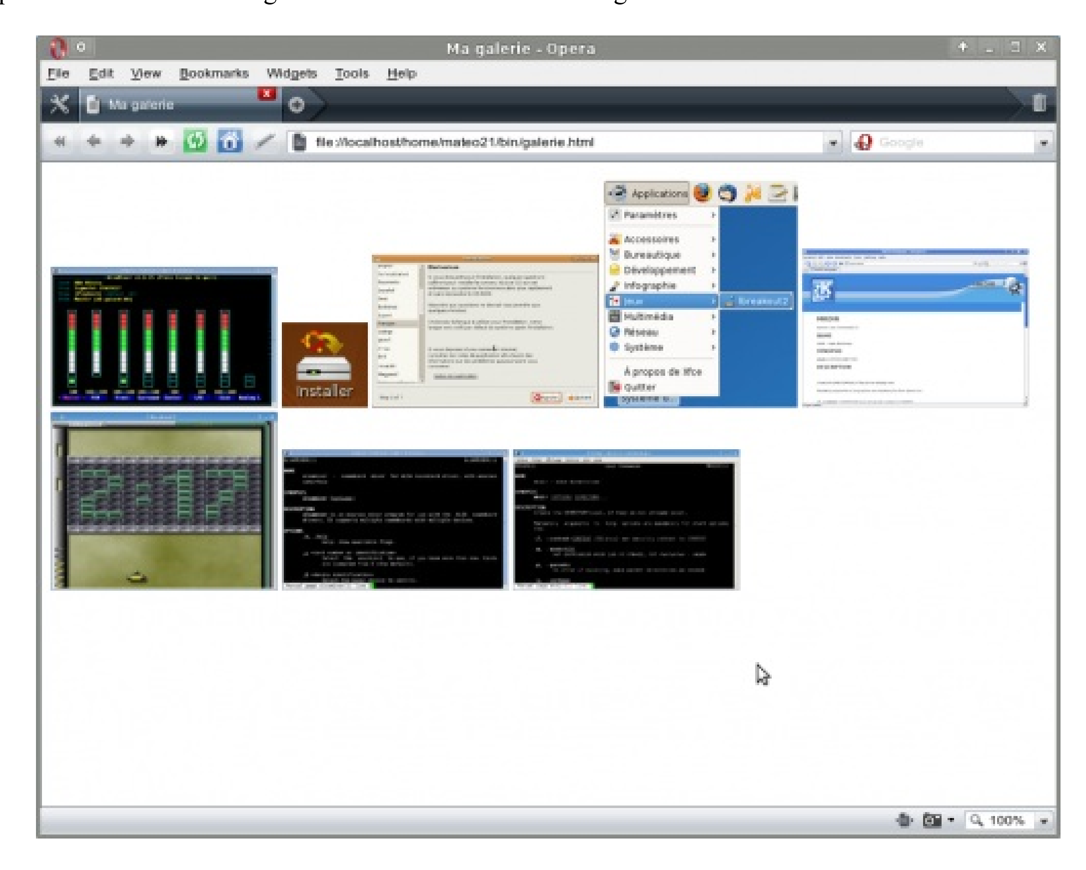
- créer une miniature de chaque image du dossier ;
- générer un fichier HTML et y insérer ces miniatures ;
- faire un lien vers les images en taille originale.



Pour réaliser ce script, il est recommandé de connaître un petit peu le HTML. Vous avez un cours à votre disposition sur le Site du Zéro. C'est assez facile : ceux qui ne connaissent pas ne devraient pas y consacrer beaucoup de temps, d'autant plus qu'il suffit de lire la première partie seulement de ce cours pour faire ce TP.

Le rendu final

La page web que vous devez arriver à générer devrait ressembler à la figure suivante.



Bien entendu, c'est un exemple **minimal**. Il est possible de faire quelque chose de beaucoup plus joli : commencez déjà par faire en sorte que cela fonctionne, vous enjoliverez après. ;)

Le code HTML de base

Pour vous aider (enfin surtout pour ceux d'entre vous qui ne sont pas très à l'aise en HTML), je vous propose de partir du code (minimaliste) suivant :

Code : Console

```
<!DOCTYPE html PUBLIC "-
//W3C//DTD XHTML 1.0 Strict//EN"  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Ma galerie</title>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
       <style type="text/css">
       a img { border:0; }
       </style>
   </head>
   <body>
      <p>

<a href="alsamixer.png"><img src="miniatures/alsamixer.png" alt="" /></a>
<a href="icone_installer.png">  <img src="miniatures/icone_installer.png" alt="" />
</a>
<a href="installation.png"><img src="miniatures/installation.png" alt="" /></a>
      </p>
   </body>
</html>
```

Vous pourrez réutiliser le début et la fin de ce code source pour chaque fichier HTML de galerie que vous créerez. Par contre, au milieu (les images `<img />` et les liens `<a>`), il faudra adapter automatiquement le code en fonction des images présentes dans le dossier.

Vous afficherez des miniatures sur la page et ferez un lien vers leur version agrandie.

Comment générer des miniatures d'images ?

Bonne question. On n'a pas appris à faire cela en ligne de commande, tout simplement parce que ce n'est pas ce que j'appelle une commande « de base » de Linux. Néanmoins, vous devriez avoir le programme `convert`, capable d'effectuer de nombreuses opérations sur des images. À vous d'afficher le manuel et de comprendre comment on l'utilise. :)

Bon : je vous aide quand même un peu parce que cette commande a énormément de paramètres. « Miniature » en anglais se dit *« thumbnail »*.
Vous apprécierez probablement l'aide en ligne, plus étoffée et plus lisible peut-être que le man.

Les paramètres

Notre programme devra accepter un paramètre optionnel : le nom du fichier HTML à générer. S'il n'est pas présent, on générera un fichier `galerie.html` par défaut.

Solution

L'heure est venue de passer à la correction !

Code : Console

```
#!/bin/bash

# Vérification des paramètres
```

```
# S'ils sont absents, on met une valeur par défaut

if [ -z $1 ]
then
        sortie='galerie.html'
else
        sortie=$1
fi

# Préparation des fichiers et dossiers

echo '' > $sortie

if [ ! -e miniatures ]
then
        mkdir miniatures
fi

# En-tête HTML

echo '<!DOCTYPE html PUBLIC "-
//W3C//DTD XHTML 1.0 Strict//EN"  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dt
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Ma galerie</title>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
       <style type="text/css">
       a img { border:0; }
       </style>
   </head>
   <body><p>' >> $sortie

# Génération des miniatures et de la page

for image in `ls *.png *.jpg *.jpeg *.gif 2>/dev/null`
do
        convert $image -thumbnail '200x200>' miniatures/$image
        echo '<a href="'$image'"><img src="miniatures/'$image'" alt="" /> </a> '>>
done

# Pied de page HTML

echo '</p>
   </body>
</html>' >> $sortie
```

Quelques commentaires sur le script :

- Il vérifie d'abord si un paramètre est présent. Si oui, il l'utilise comme nom de fichier de sortie, sinon il utilise `galerie.html`.
- On doit créer un fichier vide pour `galerie.html`. Normalement, on peut faire ça avec `touch`, mais si le fichier existe déjà, on veut le vider. On choisit donc de faire un `echo` vide dans ce fichier pour le vider.
- On crée le dossier qui accueillera les images miniatures s'il n'existe pas.
- On écrit l'en-tête HTML dans le fichier.
- On fait une boucle sur tous les fichiers de type image (*.png, *.jpg, etc.) qui existent dans le dossier. Pour chacun d'eux, on crée une miniature d'une taille maximale de 200 x 200 pixels dans le sous-dossier `miniatures`. Le petit symbole « > » permet, comme indiqué dans la documentation du programme, de ne pas générer de miniature inutilement si l'image est de base plus petite que la taille des miniatures.
- On écrit dans la page web la balise qui affichera l'image et on fait un lien vers la version agrandie.
- Enfin, on termine la page HTML en fermant les balises.

Je tiens à rappeler qu'il n'y a pas une seule façon de réaliser ce script mais plusieurs. Je vous ai présenté la mienne et, bien qu'elle fonctionne, je vous préviens que l'on peut largement l'améliorer. Je vous propose d'ailleurs des pistes pour améliorer ce script.

Améliorations

Comme je le disais plus tôt, le script que je vous ai proposé de faire est minimal. Le but était d'avoir à réaliser un script accessible

à tous et qui produise un résultat intéressant.

Si vous voulez l'améliorer, les pistes ne manquent pas. En voici quelques-unes.

- Améliorer le design de la galerie avec un peu de CSS.
- Permettre de choisir le dossier contenant les images dont on veut générer une galerie. Actuellement, il faut que `galerie.sh` soit dans le bon dossier pour que cela fonctionne !
- Utiliser un paramètre pour définir la taille des miniatures à générer.
- Afficher le nom de l'image sous chaque image.
- Afficher d'autres informations, comme les dimensions de l'image originale, sous chaque miniature. Il faudra faire appel à l'outil `convert` pour obtenir ces informations.
- Afficher la date de dernière modification sous chaque image.

Pour certaines de ces améliorations, il faudra vous renseigner dans le manuel voire poser des questions sur les forums.
Ne vous arrêtez pas en si bon chemin ! Cherchez, cherchez et cherchez encore ! Vous allez vous habituer à faire des recherches et vous deviendrez ainsi beaucoup plus autonomes.

Le cours se termine ici. Bien entendu, il aurait été possible de le compléter (presque à l'infini !) mais je ne dispose pas d'autant de temps.

Néanmoins, avec l'ensemble de ce cours vous avez désormais je l'espère une introduction à Linux enfin accessible aux débutants.
Il reste bien des choses à découvrir, je vous invite à regarder les tutoriels de la section Linux du site pour en apprendre plus si vous le souhaitez.

Bonne continuation !